TÉCNICO LISBOA

Recovery Control of Quadrotor Tailsitter UAV

Rodrigo Ken Nishikawa

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors

Dr. Meysam Basiri
Prof. Rodrigo Martins de Matos Ventura

Examination Committee

December 2024

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowlegments

First of all, I would like to thank my supervisors, Professor Rodrigo Ventura and Doctor Meysam Basiri. Their guidance has been invaluable during this dissertation.

Secondly, I am forever grateful to all my friends in São Paulo — especially Igor, Gustavo, Juliana, Luiz, Matheus, Júlia, and Caio —for the countless memories we shared together. I would also like to extend my appreciation to my friends in Lisbon, particularly Guilherme, Rodrigo, and Vinicius, who made this city feel like a second home.

Finally, I would like to thank my entire family — my mother Mônica for always encouraging me; my father Ronaldo for always believing in me; and my sister Fernanda for always inspiring me. They have supported me in every possible way, not only during this dissertation but throughout my life.

This work was supported by Aero.Next project (PRR - C645727867- 00000066) and LARSyS funding (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIDP/50009/2020 and 10.54499/UIDB/50009/2020).

Resumo

Veículos Aéreos Não Tripulados Híbridos (UAVs) combinam as capacidades de Decolagem e Aterrisagem Verticais (VTOL) e de pairar dos *rotorcrafts* com as características energeticamente eficientes de voo horizontal das aeronaves de asa fixa. Entre os diversos tipos de aeronaves híbridas, os *tailsitters* se destacam por sua simplicidade mecânica, pois realizam a transição entre o voo vertical e o voo horizontal inclinando toda a fuselagem, eliminando assim a necessidade de mecanismos complexos. Embora diversas pesquisas tenham sido realizadas focando no desempenho de *tailsitters*, particularmente durante a fase crítica de transição, há uma escassez de literatura a respeito de controle recuperação de atitude destes veículos. Esta recuperação é pertinente quando o veículo é lançado de uma outra aeronave com atitude e velocidade desconhecidas, pois aumenta a confiabilidade e segurança dos veículos não tripulados.

Esta dissertação apresenta dois controladores de recuperação distintos para um tailsitter com quatro rotores e que não possui superfícies de controle. Ambos os controladores utilizam quaterniões unitários e decompõem o erro de atitude em componentes de inclinação e direção. Um dos controladores é projetado utilizando uma abordagem de controle Proporcional-Integral-Derivativo (PID), enquanto o outro emprega um Controlador Preditivo Não Linear (NMPC) que considera todo o modelo do veículo e visa recuperar o UAV de forma otimizada, respeitando as restrições de propulsão e torque. Ambas as metodologias são avaliadas e comparadas por meio de simulações em um ambiente realístico. O PID é computacionalmente mais eficiente do que o NMPC e mais robusto a dinâmicas não modeladas. Por outro lado, para um modelo adequado, o NMPC é capaz de recuperar o UAV em menos tempo e menor perda de altitude.

Palavras chave: UAVs Híbridos, Recuperação de Atitude, Controle Preditivo Não Linear, PID.

Abstract

Hybrid Unmanned Aerial Vehicles (UAVs) integrate the Vertical Takeoff and Landing (VTOL) and hovering capabilities of rotorcrafts with the energy efficient forward flight characteristics of fixed-wing aircrafts. Among the various types of hybrid UAVs, tailsitters are notable for their mechanical simplicity, achieving transitions between hovering and forward flight by tilting the entire fuselage, eliminating the need for complex mechanisms. While extensive research has focused on the performance of tailsitters, particularly during the critical transition phase, which is characterized by substantial nonlinearities due to high angles of attack, there is limited literature addressing the recovery of tailsitters from diverse attitudes. This challenge becomes notably relevant when the vehicle is deployed from a mothership with an unknown attitude and velocity, since it can increase the reliability and safety of the tailsitters.

This dissertation provides two different recovery controllers for a quadrotor tailsitter UAV without control surfaces. Both controllers are based on unit quaternions and split the attitude error into inclination and heading components. One controller is designed using a Proportional-Integral-Derivative (PID) approach, while the other employs a Nonlinear Model Predictive Controller (NMPC) that considers the entire vehicle model and aims to recover the UAV optimally, all while adhering to the thrust and torque constraints. Both methodologies are assessed and compared through simulations in a realistic environment. The PID controller is more robust to unknown dynamics and it is more computationally efficient, while for the NMPC, for small perturbations, it is able to recover the UAV faster and with minimum altitude drop.

Keywords: Tailsitter, Attitude Recovery, Nonlinear Model Predictive Control, PID control.

Contents

List of Figures

List of Tables

List of Acronyms

AoA Angle of Attack

ARE Algebraic Riccati Equation

CFD Computational Fluid Dynamics

DCM Direction Cosine Matrix

EKF Extended Kalman Filter

LQR Linear Quadratic Regulator

MPC Model Predictive Control

NMPC Nonlinear Model Predictive Control

NED North-East-Down

PD Proportional Derivative

PID Proportional Integral and Derivative

SDF Simulation Description Format

SITL Software-in-the-Loop

UAV Unmanned Aerial Vehicle

VTOL Vertical Take-off and Landing

Nomenclature

Greek symbols

α	Angle of attack.
β	Sideslip angle.
$\boldsymbol{\omega}$	Angular velocity vector in body frame.
$\boldsymbol{\tau}$	Torque produced by rotors.
ϕ	Roll angle.
ψ	Yaw angle.
θ	Pitch angle.

Roman symbols

$\boldsymbol{q}$	Attitude quaternion from body frame to inertial frame.
$\mathbf{F}_a$	Aerodynamic Force.
$\mathbf{F}_r$	Force produced by rotors.
$\mathbf{J}$	Inertia Tensor.
$\mathbf{L}, \mathbf{M}, \mathbf{N}$	Rolling, Pitching, Yawing Moments.
$\mathbf{M}_a$	Aerodynamic Moment.
$\mathbf{p}$	Position vector in inertial frame.
$\mathbf{R}$	Rotation Matrix from body frame to inertial frame.
$\mathbf{u}$	Control input vector.

$\mathbf{v}$	Velocity vector in inertial frame.
$\mathbf{w}$	Wind speed in inertial frame.
$\mathcal{B}$	Body frame.
$\mathcal{I}$	Inertial Frame.
$\mathcal{L}, \mathcal{D}, \mathcal{Y}$	Lift, Drag, Side Force.
C_L, C_D, C_Y	Lift, Drag, Side force Coefficients.
C_l, C_m, C_n	Rolling, Pitching, Yawing Moments Coefficients.
g	Gravitational Acceleration.
J	Objective Function.
m	Mass of UAV.
V	Magnitude of airspeed.
T	Thrust force.

Superscripts

err	Error.
ref	Reference.
T	Transpose.

Subscripts

x, y, z	Cartesian components.
d	Desired.
max	Maximum.
min	Minimum.

Chapter 1

Introduction

In recent years, Unmanned Aerial Vehicles (UAVs) have experienced tremendous development, as they can be used in a wide range of applications, such as surveillance [1], supplies transportation [2], search and rescue [3], and environmental monitoring [4], to name a few. Within different UAV types, rotorcraft and fixed-wing designs are still the most commonly used. Fixed-wing UAVs are more energy efficient and possess a substantial payload capacity. Nevertheless, they require dedicated runways or catapult systems for takeoff and landing and need to stay in motion to generate the required lift. Conversely, rotorcrafts have Vertical Take-off and Landing (VTOL) capabilities, are able to hover, and have superior maneuverability, making them ideal for operations requiring high precision or confined space navigation. However, they are less energy efficient, slower, and have restricted payload capacity and endurance.

Despite their prevalence, both rotorcrafts and fixed-wing UAVs come with inherent limitations. To harness the advantages of both types, a new of breed of vehicle - the hybrid UAV - was developed. Hybrid UAVs integrate VTOL capabilities along with energy efficient flight, enabling a broader spectrum of applications. Nonetheless, their complexity presents added challenges in terms of vehicle modeling and control, particularly during transitions between flight modes. The need for seamless mode-switching without compromising stability demands sophisticated control algorithms capable of adapting to dynamic changes in flight conditions.

Hybrid UAVs can generally be divided into different categories, such as tiltrotors, tiltwings, extra-propulsion, and tailsitters [5]. A tiltrotor UAV has rotors that can be tilted by a servo motor to provide either the thrust during hovering and vertical takeoff and landing or lift during forward flight, as seen in Fig. 1.1b. A tiltwing, shown in

Fig. 1.1c, has a transition mechanism similar to a tiltrotor, except that the wings also tilt instead of the rotors alone. The extra-propulsion UAV has different rotors for vertical and forward flight, as seen in Fig. 1.1d. As a consequence, their design, modeling, and control are relatively simple. However, in horizontal flight, the multiple non-operational motors used during takeoff cause extra drag due to their fixed position, resulting in additional burden to the tractors or pushers [6]. Finally, tailsitters UAVs are characterized by having a relatively simple mechanical structure, as they do not require moving parts to transition between hovering and cruise flight. Instead, the entire airframe rotates, aligning its lift to counteract gravity and redirecting thrust forward for propulsion. While this simplicity in design reduces mechanical complexity, it increases the demands on the control system, particularly due to the nonlinearity of the aerodynamics at high Angles of Attack (Angle of Attack (AoA)s) during the transition. An example of tailsitter is seen in Fig. 1.1a.

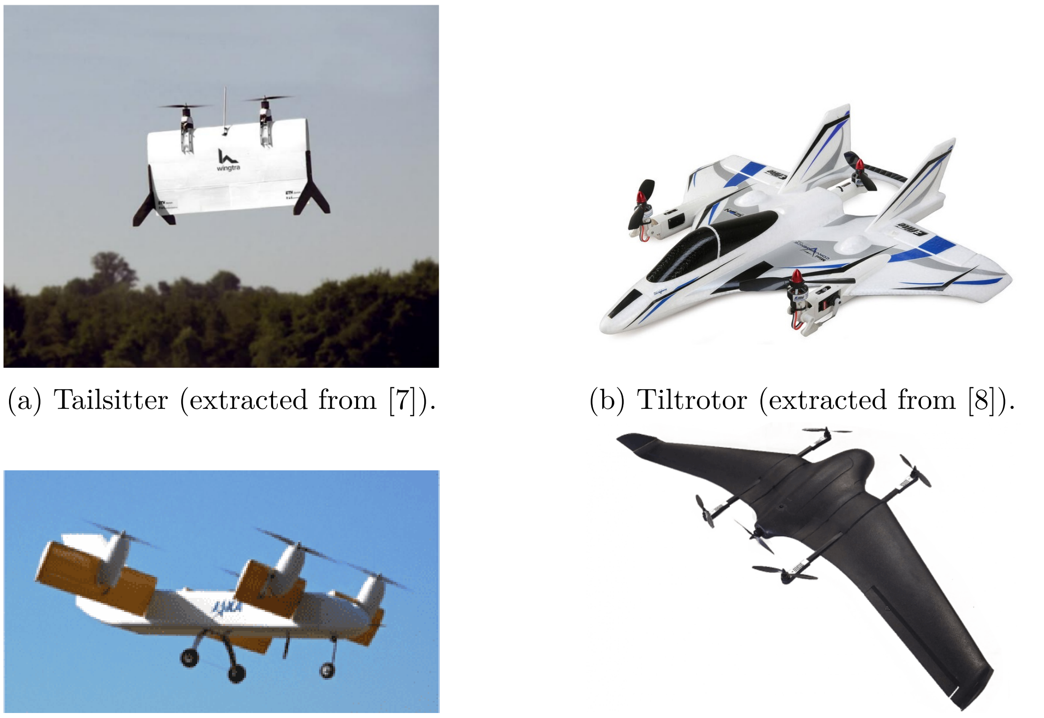

(a) Tailsitter (extracted from [7]).
(b) Tiltrotor (extracted from [8]).
(c) Tiltwing (extracted from [9]).
(d) Dual-system (extracted from [8]).

Figure 1.1: Examples of hybrid UAVs.

The concept of deploying UAVs from a mothership introduces significant operational advantages that are essential in extending the capabilities of UAV missions. By using a mothership as a mobile launch platform, UAVs can be transported closer to their intended operational areas, conserving energy and increasing their effective range. This is particularly valuable for small UAVs with limited fuel or battery capacity, enabling them to perform longer missions without the need for extensive ground infrastructure or risky long-distance flights. In scenarios where ground takeoff is impractical—such as over oceans, mountainous regions, or in hostile environments—a mothership provides the flexibility to launch UAVs in difficult-to-reach areas while staying out of danger zones.

Moreover, mothership deployment enhances mission versatility by supporting coordinated multi-UAV operations. A mothership can carry several UAVs and deploy them either simultaneously or in a phased manner, allowing for broader coverage or multiple operational roles, such as reconnaissance, surveillance, or target engagement. This capability is particularly beneficial in military or disaster response scenarios where rapid deployment and real-time adjustments are critical.

The strategic benefits of mothership deployment also include improving stealth and reducing the risk of detection. By keeping the mothership far from sensitive or contested areas, UAVs can operate covertly, increasing the likelihood of mission success while minimizing risk to the primary vessel. This makes mothership deployment invaluable for reconnaissance or surveillance missions that require a low-profile approach.

However, after deployment from a mothership aircraft, a critical challenge is the rapid stabilization of the UAV. As soon as the vehicle is released, it must transition from its deployment state to stable flight, either by hovering or moving into forward flight. The development of an effective stabilization or recovery controller is therefore key to ensure mission success.

1.1 Motivation

In this thesis, the focus will be on a specific configuration of hybrid UAV: a quadrotor tailsitter without control surfaces, similar to the commercially available Swan-K1 [1], as seen in Fig. 1.2. This tailsitter UAV presents four rotors for full attitude control and thrust generation for both vertical and forward flight. Since it does not have control surfaces, its actuation principle is similar to a quadrotor UAV. This particular model presents four rotors are slightly inclined, providing more control authority.

When dealing with tailsitters, one of the main concerns is the control of these aircrafts, which present highly nonlinear and difficult to model dynamics. Particularly, since the whole fuselage tilts to achieve level flight, the wings operate at large AoA, which increases

[1]Holybro, Swan k1, https://holybro.com/products/swan-k1, Accessed: August 2024.

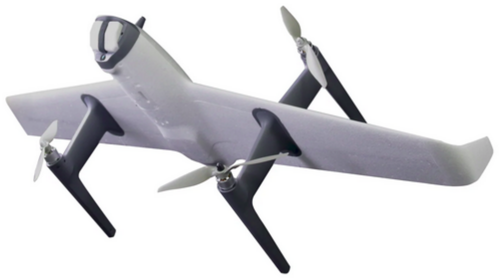


Figure 1.2: Swan-K1 quadrotor tailsitter (extracted from [1]).

the complexity of the aerodynamics.

A standard approach to control hybrid UAVs is to divide the flight envelope into three different modes: VTOL, transition, and cruise flight. Different controllers are then designed for each of these phases. Sometimes the transition is neglected and only two modes are defined, as in the case of the PX4 autopilot [8]. In [10], a fourth mode, called recovery mode, was introduced for a tailsitter UAV to handle situations where the vehicle enters unexpected flight states. An alternative is to design a single flight controller to deal with the full flight envelope for hybrid UAVs as in [11].

To enhance the autonomy and safety of UAVs, it is crucial to design a recovery controller capable of stabilizing the vehicle when it enters unexpected flight states. Of particular importance in the scope of this thesis is the free-falling scenario, which occurs when the UAV is dropped from a larger aircraft. However, much of the existing literature on hybrid UAVs focuses on well-defined flight conditions where the vehicle remains under control. For tailsitters, in particular, the emphasis is often on the transition strategy from hovering to level flight, as well as the back transition from level flight to hovering. Moreover, research on recovery procedures for hybrid UAVs is rather limited, with most studies centered around rotorcrafts rather than hybrid designs. Thus, the primary objective of this work is to develop a recovery controller applied to a tailsitter UAV.

1.2 Objectives

The main objective of this thesis is to design a recovery controller that stabilizes a quadrotor tailsitter UAV by leveraging the passive stabilization offered by the vehicle's wings to enhance efficiency during recovery. The work will involve the implementation and testing of different methodologies to evaluate the effectiveness of the proposed recovery strategies.

1.3 Problem Statement

The challenge addressed in this thesis is the stabilization of a quadrotor tailsitter UAV without control surfaces. The primary objective is to design a recovery controller capable of stabilizing the vehicle in scenarios where it experiences free-falls or loses control during aggressive maneuvers. The controller must enable the vehicle to automatically recover from arbitrary initial attitude and velocities. The vehicle is considered recovered when its attitude is such that the vehicle is pointing up and its height is stable. In Fig. 1.3, the desired attitude is represented when the angle ϑ is equal to zero rad. For the implementation of the recovery controller, it is assumed that a state estimator is available, providing information on the aircraft's position, attitude, angular and linear velocities.

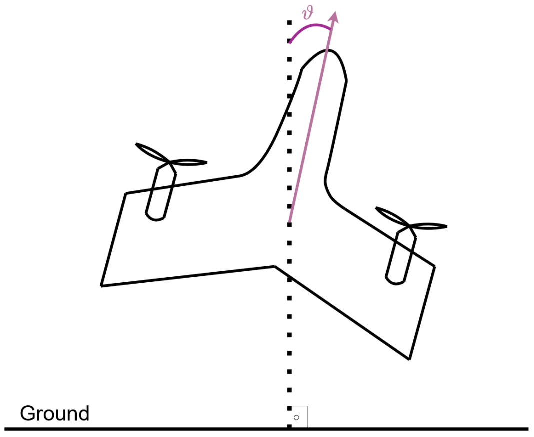


Figure 1.3: Representation of the inclination of the vehicle with respect to the ground. The vehicle is pointing up when ϑ is equal to zero rad.

1.4 Contribution

The problem posed in the previous section is addressed by designing two different control strategies. The first approach involves the implementation of two PID controllers, one for the attitude and another for the altitude. Feedforward terms on the aerodynamic forces and moments are introduced in the design of the PIDs. The second approach consists of developing an unified NMPC specifically for the recovery of the tailsitter UAV, leveraging the full nonlinear dynamics of the vehicle and including constraints on the thrust and torques.

Both methodologies use unit quaternions to represent the attitude and decompose the attitude error into inclination and heading components. The outputs of the recovery controllers are the desired thrust and torques, which are sent to the flight controller, running in offboard mode. Software-in-the-Loop (SITL) simulations were conducted to evaluate and compare the controllers. The code documentation used throughout this dissertation can be found in [2].

1.5 Thesis Outline

The remainder of the thesis is structured as follows: Chapter 2 provides the theoretical background on the necessary concepts to consider throughout this dissertation. Chapter 3 presents a discussion on the state of the art regarding recovery control of UAVs, attitude control of tailsitter, and Predictive Control in deep stall. In Chapter 4, a model of the tailsitter considered is derived and the simulation environment is detailed. Chapter 5 describes the control architecture and the implementation of the two different controllers, while Chapter 6 presents the simulation results obtained. Finally, Chapter 7 summarizes the work performed and highlights the main achievements in this work. Moreover, this chapter proposes further work to extend the activities described in this document.

[2] `https://github.com/rknishi/recovery_quadtailsitter.git`

Chapter 2

Background

This chapter provides an overview on attitude representation, kinematics, and dynamics. Additionally, it presents the fundamental theory behind the two controllers used in this thesis: Model Predictive Control (MPC) and Proportional Integral and Derivative (PID) control.

2.1 Attitude Representation

It is desirable to represent the three-dimensional (3D) orientation of a moving-frame $\mathcal{B} : \{\mathbf{O}_{\mathcal{B}}, \hat{\mathbf{i}}_{\mathcal{B}}, \hat{\mathbf{j}}_{\mathcal{B}}, \hat{\mathbf{k}}_{\mathcal{B}}\}$ attached to a rigid body with respect to a fixed inertial frame $\mathcal{I} : \{\mathbf{O}_{\mathcal{I}}, \hat{\mathbf{i}}_{\mathcal{I}}, \hat{\mathbf{j}}_{\mathcal{I}}, \hat{\mathbf{k}}_{\mathcal{I}}\}$. Both coordinates frames are defined based on their origin $\mathbf{O}$ and a set of three mutually orthogonal unit direction vectors $\{\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}\}$. More details on these frames are provided in Sec. 4.

The configuration manifold of the attitude dynamics of a rigid body is the special orthogonal group:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} : \mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \det \mathbf{R} = 1\} \tag{2.1}$$

where $\mathbf{I}_3$ is the identity matrix with size 3×3 and $\mathbf{R} \in SO(3)$ is called a rotation matrix. Attitude control is typically studied using various attitude parameterizations, such as Euler Angles, Angle-Axis, and Unit Quaternions.

All parameterization methods serve the same purpose of attitude representation, but each has its own advantages and disadvantages such as the number of parameters required or existing singularities. A brief introduction to them will be given in this section, that was based on [12, 13, 14, 15, 16].

2.1.1 Rotation Matrices

The attitude of a rigid body can be modeled by a linear transformation between a reference frame and a body-fixed frame that preserves the distance between each pair of material points in the body and the handedness of coordinate frames. A rotation matrix is a matrix whose multiplication with a vector rotates the vector while preserving its length and can be used to represent the attitude of a rigid body. Let $\mathbf{p}$ describe a generic vector, $\mathcal{B}$ and $\mathcal{I}$ be two distinct reference frames, specified by the set of perpendicular unit axis $(\hat{\mathbf{i}}_{\mathcal{B}}, \hat{\mathbf{j}}_{\mathcal{B}}, \hat{\mathbf{k}}_{\mathcal{B}})$ and $(\hat{\mathbf{i}}_{\mathcal{I}}, \hat{\mathbf{j}}_{\mathcal{I}}, \hat{\mathbf{k}}_{\mathcal{I}})$, respectively and origins $\mathbf{O}_{\mathcal{B}}$ and $\mathbf{O}_{\mathcal{I}}$. The vector $\mathbf{p}$ can be expressed in both frames $\mathcal{B}$ and $\mathcal{I}$ as $\mathbf{p}^{\mathcal{B}}$ and $\mathbf{p}^{\mathcal{I}}$, respectively. For the former:

$$\mathbf{p}^{\mathcal{B}} = p_x^{\mathcal{B}}\,\hat{\mathbf{i}}_{\mathcal{B}} + p_y^{\mathcal{B}}\,\hat{\mathbf{j}}_{\mathcal{B}} + p_z^{\mathcal{B}}\,\hat{\mathbf{k}}_{\mathcal{B}}. \tag{2.2}$$

And for the latter:

$$\mathbf{p}^{\mathcal{I}} = p_x^{\mathcal{I}}\,\hat{\mathbf{i}}_{\mathcal{I}} + p_y^{\mathcal{I}}\,\hat{\mathbf{j}}_{\mathcal{I}} + p_z^{\mathcal{I}}\,\hat{\mathbf{k}}_{\mathcal{I}}. \tag{2.3}$$

Setting the two expressions equal to each other gives:

$$\mathbf{p} = p_x^{\mathcal{I}}\,\hat{\mathbf{i}}_{\mathcal{I}} + p_y^{\mathcal{I}}\,\hat{\mathbf{j}}^{\mathcal{I}} + p_z^{\mathcal{I}}\,\hat{\mathbf{k}}_{\mathcal{I}} = p_x^{\mathcal{B}}\,\hat{\mathbf{i}}_{\mathcal{B}} + p_y^{\mathcal{B}}\,\hat{\mathbf{j}}_{\mathcal{B}} + p_z^{\mathcal{B}}\,\hat{\mathbf{k}}_{\mathcal{B}}. \tag{2.4}$$

Then, assuming both frames are orthonormal and with the same origin, taking the dot product of both sides with $\hat{\mathbf{i}}^{\mathcal{I}}$, $\hat{\mathbf{j}}^{\mathcal{I}}$, and $\hat{\mathbf{k}}^{\mathcal{I}}$ respectively, and stacking the result into matrix form results:

$$\mathbf{p} \triangleq \begin{bmatrix} p_x^{\mathcal{I}} \\ p_y^{\mathcal{I}} \\ p_z^{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{i}}_{\mathcal{I}} \cdot \hat{\mathbf{i}}_{\mathcal{B}} & \hat{\mathbf{i}}_{\mathcal{I}} \cdot \hat{\mathbf{j}}_{\mathcal{B}} & \hat{\mathbf{i}}_{\mathcal{I}} \cdot \hat{\mathbf{k}}_{\mathcal{B}} \\ \hat{\mathbf{j}}_{\mathcal{I}} \cdot \hat{\mathbf{i}}_{\mathcal{B}} & \hat{\mathbf{j}}_{\mathcal{I}} \cdot \hat{\mathbf{j}}_{\mathcal{B}} & \hat{\mathbf{j}}_{\mathcal{I}} \cdot \hat{\mathbf{k}}_{\mathcal{B}} \\ \hat{\mathbf{k}}_{\mathcal{I}} \cdot \hat{\mathbf{i}}_{\mathcal{B}} & \hat{\mathbf{k}}_{\mathcal{I}} \cdot \hat{\mathbf{j}}_{\mathcal{B}} & \hat{\mathbf{k}}_{\mathcal{I}} \cdot \hat{\mathbf{k}}_{\mathcal{B}} \end{bmatrix} \begin{bmatrix} p_x^{\mathcal{B}} \\ p_y^{\mathcal{B}} \\ p_z^{\mathcal{B}} \end{bmatrix} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\,\mathbf{p}^{\mathcal{B}}. \tag{2.5}$$

The matrix ${}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}$ represents the rotation matrix from frame $\mathcal{B}$ to frame $\mathcal{I}$. Hence, the attitude is represented by a 3×3 matrix, describing the rotation between two reference frames. Since the elements of a rotation matrix are the cosines of the angles between the two sets of axes, the rotation matrix is also referred to as Direction Cosine Matrix (DCM). Usually, a simplified notation is used: $\mathbf{R} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}$

Notice that a generic 3×3 matrix contains nine elements, as three generic $\mathbb{R}^3$ vectors have nine independent components. However, as the vectors are unitary, the number of independent components drop to six. Additionally, as those vectors are mutually perpendicular, with null projection, three constraints are added such that the number of independent components is dropped to only three. Since the basis of a reference frame is orthonormal, the DCM belongs to this special condition, and only three independent components are necessary to fully express a DCM.

2.1.2 Euler Angles and Tait-Bryan angles

A coordinate rotation is a rotation about a single coordinate axis. Three coordinate rotations in sequence around three non-consecutive axes can describe any rotation in three-dimensional space, which leads to an intuitive and simple way to represent the attitude of a rigid body. The rotations may be intrinsic, in which the rotation axis changes solidary with the the moving body, or extrinsic, rotations around the axes of an inertial reference frame.

The parameterization consists of a vector of three angles $\begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^T$, where each angle represents a rotation about an axis of one frame in relation to the other frame. The sequence of rotations is important and a total of twelve different sets to express the attitude can be defined.

One of the most common choices of rotation sequence is the Z-Y-X, also known as the yaw ψ, pitch θ, roll ϕ. The body frame is initially coincident with the inertial frame. There is a first rotation of ψ (yaw) around the z-axis of the original body frame, then a rotation of θ (pitch) around the new y-axis of the body, and finally a rotation of ϕ (yaw) about the newest x-axis. Each of these coordinate rotation may be written in the form of rotation matrices:

$$\begin{aligned}
\mathbf{R_x}(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \\
\mathbf{R}_y(\theta) &= \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} . \\
\mathbf{R}_z(\psi) &= \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.6}$$

The rotation matrix $\mathbf{R}$ from the body frame $\mathcal{B}$ to the inertial frame $\mathcal{I}$ is then obtained by the multiplication of the matrices $\mathbf{R}_z(\psi)$, $\mathbf{R}_y(\theta)$, and $\mathbf{R}_x(\phi)$:

$$\mathbf{R}(\psi,\theta,\phi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix}. \tag{2.7}$$

Extrinsic rotations follow the same rules and mathematical formulations of intrinsic ones. The difference is the naming of the coordinate rotation and the association with the axes. Any extrinsic rotation is equivalent to an intrinsic rotation by the same angles, but with inverted order of elemental rotations and vice-versa.

The Z-Y-X Euler angles can be retrieved by:

$$\theta = \arcsin(-\mathbf{R}_{31}), \tag{2.8}$$

$$\psi = \arctan 2(\mathbf{R}_{21}, \mathbf{R}_{11}), \tag{2.9}$$

$$\phi = \arctan 2(\mathbf{R}_{32}, \mathbf{R}_{33}), \tag{2.10}$$

where the subscript ij represents the element in i-th line and j-th column of $\mathbf{R}$.

As will be discussed in Sec. 2.2.1, a well known problem of Euler angles is the existence of singularities. For the Z-Y-X parameterization, this singularity occurs at $\theta = \pm\frac{\pi}{2}$.

Considering that the focus of this dissertation is on a tailsitter, which tilts its pitch angle by almost 90° in the flight envelope, a more convenient representation is the Z-X-Y intrinsic Euler angles [5]. In this case, there is a first rotation of ψ (yaw) around the original z-axis, then a second rotation of ϕ (roll) around the new x-axis, and finally a third rotation of θ (pitch) around the newest y-axis. Notice that this is the same as the Y-X-Z extrinsic rotations, where you first rotate about the original Y of the original frame by θ, then about the original fixed frame X axis by ϕ, and finally by the original Z axis by ψ. With that, the rotation matrix is given by:

$$\mathbf{R}(\theta,\phi,\psi) = \mathbf{R}_Z(\psi)\mathbf{R}_X(\phi)\mathbf{R}_Y(\theta) = \begin{bmatrix} c_\psi c_\theta - s_\phi s_\psi s_\theta & -s_\psi c_\phi & s_\theta c_\psi + s_\phi s_\psi c_\theta \\ s_\psi c_\theta + s_\phi s_\theta c_\psi & c_\phi c_\psi & s_\theta s_\psi - s_\phi c_\psi c_\theta \\ -s_\theta c_\phi & s_\phi & c_\phi c_\theta \end{bmatrix}. \tag{2.11}$$

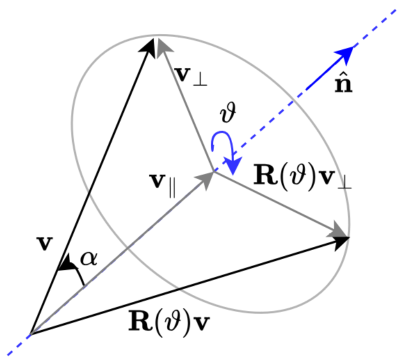


Figure 2.1: Rotation of a vector $\mathbf{v}$ around $\hat{\mathbf{n}}$ by an angle ϑ. Adapted from [16].

Then, the Euler Angles for the Z-X-Y parameterization can be calculated by:

$$\phi = \arcsin(\mathbf{R_{32}}), \tag{2.12}$$

$$\theta = \arctan 2(-\mathbf{R}_{31}, \mathbf{R}_{33}), \tag{2.13}$$

$$\psi = \arctan 2(-\mathbf{R}_{12}, \mathbf{R}_{22}), \tag{2.14}$$

with that, the singular point shifts from $\theta = \pm\frac{\pi}{2}$ to $\phi = \pm\frac{\pi}{2}$.

2.1.3 Axis-Angle representation

The axis-angle represents a rotation matrix $\mathbf{R}_{\hat{\mathbf{n}}}(\vartheta)$ as a rotation of a given angle ϑ about a given unit axis $\hat{\mathbf{n}}$.

Let $\mathbf{v}$ be an arbitrary vector, $\hat{\mathbf{n}}$ an arbitrary unit vector, and ϑ the angle of rotation of $\mathbf{v}$ around $\hat{\mathbf{n}}$. Decomposing $\mathbf{v}$ into a part $\mathbf{v}_\perp$ perpendicular to $\hat{\mathbf{n}}$ and a part $\mathbf{v}_\parallel$ parallel to $\hat{\mathbf{n}}$:

$$\mathbf{v}_\parallel = \hat{\mathbf{n}}(||\mathbf{v}|| \cos\alpha) = \hat{\mathbf{n}}\hat{\mathbf{n}}^T\mathbf{v}, \tag{2.15}$$

$$\mathbf{v}_\perp = \mathbf{v} - \mathbf{v}_\parallel = \mathbf{v} - \hat{\mathbf{n}}\hat{\mathbf{n}}^T\mathbf{v}, \tag{2.16}$$

where α is the angle between $\mathbf{v}$ and $\hat{\mathbf{n}}$, as illustrated in Fig. 2.1.

The parallel part does not rotate upon rotation around $\hat{\mathbf{n}}$:

$$\mathbf{R}_{\hat{\mathbf{n}}}(\vartheta)\,\mathbf{v}_\parallel = \mathbf{v}_\parallel, \tag{2.17}$$

while the orthogonal part rotates in the plane normal to $\hat{\mathbf{n}}$:

$$\mathbf{R}_{\hat{\mathbf{n}}}(\vartheta)\,\mathbf{v}_\perp = \mathbf{v}_\perp \cos\vartheta + (\hat{\mathbf{n}} \times \mathbf{v})\sin\vartheta. \tag{2.18}$$

Therefore, in general,

$$\mathbf{R}_{\hat{\mathbf{n}}}(\vartheta)\,\mathbf{v} = \mathbf{v}_\parallel + \cos\vartheta\mathbf{v}_\perp + (\hat{\mathbf{n}} \times \mathbf{v})\sin\vartheta, \tag{2.19}$$

which is known as the vector rotation formula.

By using the Rodrigues' rotation formula, the rotation matrix $\mathbf{R}_{\hat{\mathbf{n}}}(\vartheta)$ can be computed as:

$$\mathbf{R}_{\hat{\mathbf{n}}}(\vartheta) = \mathbf{I}_3 + \sin\vartheta S(\hat{\mathbf{n}}) + (1-\cos\vartheta)S(\hat{\mathbf{n}})^2, \tag{2.20}$$

where $\mathbf{I}_3$ is the identity matrix and $S(.)$ is the skew-symmetric operator such that $\mathbf{a}\times\mathbf{b} = S(\mathbf{a})\mathbf{b}$.

The angle ϑ and axis $\hat{\mathbf{n}}$ of rotation can be recovered from the rotation matrix $\mathbf{R}$ according to:

$$\cos\vartheta = \frac{1}{2}(tr(\mathbf{R}) - 1), \tag{2.21}$$

where $tr(\mathbf{R})$ indicates the trace of $\mathbf{R}$. For $\sin\vartheta \neq 0$:

$$\hat{\mathbf{n}} = \frac{1}{2\sin\vartheta}\begin{bmatrix} \mathbf{R}_{23} - \mathbf{R}_{32} \\ \mathbf{R}_{31} - \mathbf{R}_{13} \\ \mathbf{R}_{12} - \mathbf{R}_{21} \end{bmatrix}. \tag{2.22}$$

For $\vartheta = 0$, $\hat{\mathbf{n}}$ is undefined but not physically significant [14]. For $\vartheta = \pi$, $\mathbf{R}$ has the form:

$$\mathbf{R}_{\hat{\mathbf{n}}}(\pi) = -\mathbf{I}_3 + 2\hat{\mathbf{n}}\hat{\mathbf{n}}^T, \tag{2.23}$$

so that any column of $\mathbf{I}_3 + \mathbf{R}_{\hat{\mathbf{n}}}(\pi)$ is proportional to $\hat{\mathbf{n}}$. The sign of $\hat{\mathbf{n}}$ is not significant in this case [15]. It is remarked that the direct transformation, from axis-angle to the attitude matrix, is always possible, while the inverse transformation is not always defined [15].

2.1.4 Quaternions

In its most general form, a quaternion $Q \in \mathbb{H}$ is an extended number system for complex numbers, where $\mathbb{H}$ is the Hamilton space. A quaternion Q can be represented by:

$$Q = a + bi + cj + dk \in \mathbb{H}, \tag{2.24}$$

where $a, b, c, d \in \mathbb{R}$ and i, j, k are three imaginary unit numbers defined such that:

$$i^2 = j^2 = k^2 = -1. \tag{2.25}$$

A quaternion Q can be posed as a sum of a scalar part q_0 and a vector part $\tilde{\mathbf{q}}$:

$$Q = q_0 + q_1\, i + q_2 j + q_3\, k \Leftrightarrow Q = q_0 + \tilde{\mathbf{q}}, \tag{2.26}$$

where q_0 is referred to as the real or scalar part and $\tilde{\mathbf{q}} = q_1\, i + q_2 j + q_3\, k = (q_1, q_2, q_3)$ as the imaginary or vector part.

A quaternion can be represented as a 4-vector $\boldsymbol{q}$:

$$\boldsymbol{q} \triangleq \begin{bmatrix} q_0 \\ \tilde{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\frac{\vartheta}{2} \\ \sin\frac{\vartheta}{2}\hat{\mathbf{n}} \end{bmatrix}, \tag{2.27}$$

which allows the use of matrix algebra for operations using quaternions.

The norm of a quaternion $\boldsymbol{q}$ is given by:

$$\|\boldsymbol{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \in \mathbb{R}. \tag{2.28}$$

The quaternion conjugate $\overline{\boldsymbol{q}}$ is given by:

$$\overline{\boldsymbol{q}} = \begin{bmatrix} q_0 \\ -\tilde{\mathbf{q}} \end{bmatrix}. \tag{2.29}$$

Quaternion multiplication is not cummutative. Quaternion multiplication between two quaternions $\boldsymbol{p} = (p_0, \mathbf{p})$ and $\boldsymbol{q} = (q_0, \tilde{\mathbf{q}})$ is defined by

$$\boldsymbol{p} \circ \boldsymbol{q} = \begin{bmatrix} p_0 q_0 - \mathbf{p}^T \tilde{\mathbf{q}} \\ p_0\, \tilde{\mathbf{q}} + q_o\, \mathbf{p} + \mathbf{p} \times \tilde{\mathbf{q}} \end{bmatrix}. \tag{2.30}$$

When dealing with three-dimensional rotations, the scope is limited to unit quaternions, that is $\|\boldsymbol{q}\| = 1$. Unlike Euler angles, quaternion representation does not have singularities. However, they double cover the set of attitudes $SO(3)$, meaning that each attitude corresponds to two different quaternion vectors. Specifically, a physical attitude $\mathbf{R} \in SO(3)$ is represented by a pair of antipodal quaternions $\pm\boldsymbol{q}$. In other words, for a given rotation matrix, there is ambiguity regarding the corresponding quaternion, as both $+\boldsymbol{q}$ and $-\boldsymbol{q}$ map to that matrix.

The unit quaternion can be interpret as a single rotation about an axis in three-dimensional space, which according to Euler's Theorem, can describe any 3D rotation. A rotation of ϑ about the unit axis $\hat{\mathbf{n}} = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}^T$ can be represented by

$$\boldsymbol{q} = \begin{bmatrix} \cos\frac{\vartheta}{2} \\ \sin\frac{\vartheta}{2}\hat{\mathbf{n}} \end{bmatrix}. \tag{2.31}$$

Once again, let $\mathbf{v}$ be a generic vector, $\mathcal{B}$ and $\mathcal{I}$ two different reference frames and $\boldsymbol{q}_{\mathcal{IB}}$ the quaternion representing the orientation of frame $\mathcal{I}$ with respect to frame $\mathcal{B}$. To simplify the notation, $\boldsymbol{q}_{\mathcal{IB}}$ is usually written simply as $\boldsymbol{q}$. Vector $\mathbf{v}$ can be expressed relative to frame $\mathcal{I}$ by considering $\boldsymbol{v} = (0, \mathbf{v})$ and performing:

$$\boldsymbol{v}^{\mathcal{I}} = \boldsymbol{q} \circ \boldsymbol{v}^{\mathcal{B}} \circ \overline{\boldsymbol{q}} = (0, \mathbf{v}^{\mathcal{I}}) = (0, \mathbf{R}(\boldsymbol{q})\mathbf{v}^{\mathcal{B}}) = \boldsymbol{q} \otimes \mathbf{v}^{B}, \tag{2.32}$$

where the matrix $\mathbf{R}(\boldsymbol{q})$ is

$$\mathbf{R}(\boldsymbol{q}) = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1 \end{bmatrix}. \tag{2.33}$$

It follows that both $+\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation matrix. If $\boldsymbol{q}$ is the quaternion that represents the orientation of $\mathcal{I}$ relative to $\mathcal{B}$, then $\mathbf{R}(\boldsymbol{q})$ corresponds to $\mathbf{R}$.

A rotation of $\boldsymbol{q}_a$ followed by a rotation of $\boldsymbol{q}_b$ can be combined into a single unit quaternion $\boldsymbol{q}_c$ as:

$$\boldsymbol{q}_c = \boldsymbol{q}_b \circ \boldsymbol{q}_a. \tag{2.34}$$

To represent the relative quaternion $\boldsymbol{\delta q}$ between an attitude state $\boldsymbol{q}$ and a reference attitude $\boldsymbol{q}_{ref}$, both represented by two quaternions, quaternion multiplication can be used:

$$\boldsymbol{\delta q} = \boldsymbol{q}_{ref} \circ \boldsymbol{q}^{-1}, \tag{2.35}$$

which is valid for arbitrarily large rotations. Considering unit quaternions, the inverse $\boldsymbol{q}^{-1}$ is the same as the conjugate $\overline{\boldsymbol{q}}$.

2.2 Attitude Kinematics

The angular motion of $\mathcal{B}$ relative to $\mathcal{I}$ is described through the angular velocity vector $\boldsymbol{\omega}_{\mathcal{IB}}$:

$$\boldsymbol{\omega}_{\mathcal{IB}} = \omega_x\hat{\mathbf{i}} + \omega_y\hat{\mathbf{j}} + \omega_z\hat{\mathbf{k}}. \tag{2.36}$$

This vector is the instantaneous angular rotation vector of body $\mathcal{B}$ relative to $\mathcal{I}$ and is typically expressed with respect to the body frame as $\boldsymbol{\omega}^{\mathcal{B}}_{\mathcal{IB}}$. If only $\mathcal{B}$ and $\mathcal{I}$ frames are considered, then $\boldsymbol{\omega}^{\mathcal{B}}_{\mathcal{IB}}$ is often written simply as $\boldsymbol{\omega}$.

It is also possible to express the angular velocity vector in the inertial frame as:

$$\boldsymbol{\omega}^{\mathcal{I}}(t) = \mathbf{R}(t)\boldsymbol{\omega}(t). \tag{2.37}$$

The relation between the two vectors is given by the DCM whose variation in time is a function of the angular velocity itself.

The fundamental kinematics relation can be written in terms of the angular velocity expressed in the body frame as:

$$\dot{\mathbf{R}} = \mathbf{R}S(\boldsymbol{\omega}) \tag{2.38}$$

or, with the angular velocity expressed in the inertial frame:

$$\dot{\mathbf{R}} = S(\boldsymbol{\omega}^{\mathcal{I}})\mathbf{R}. \tag{2.39}$$

2.2.1 Euler angles kinematics

The variation in time of the Euler angles can be related to the angular velocity of the body by differentiation of the Euler angles relations, which for the Z-Y-X sequence is:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.40}$$

When $\theta \pm \frac{\pi}{2}$, the derivative is infinite. This condition is related to the gimbal lock phenomenon.

For the Z-X-Y sequence:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \tan\phi\sin\theta & 1 & -\tan\phi\cos\theta \\ -\frac{\sin\theta}{\cos\phi} & 0 & \frac{\cos\theta}{\cos\phi} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{2.41}$$

In this case, the singularity is at $\phi \pm \frac{\pi}{2}$.

2.2.2 Quaternions Kinematics

Unlike Euler angles, quaternions provide a singularity-free relation between quaternion derivative and angular velocity of the reference frame:

$$\dot{\boldsymbol{q}} = \frac{1}{2} \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -S(\boldsymbol{\omega}) \end{bmatrix} \boldsymbol{q} \tag{2.42}$$

which does not present singularities.

To integrate the quaternion kinematics, it is often more convenient to use a different form:

$$\dot{\boldsymbol{q}} = \frac{1}{2} \begin{bmatrix} -\tilde{\mathbf{q}}^T \\ \mathbf{I}_3 q_0 + S(\tilde{\mathbf{q}}) \end{bmatrix} \boldsymbol{\omega}, \tag{2.43}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix.

2.3 Attitude Dynamics

Angular momentum $\mathbf{h}$ is a vector quantity that connects the angular velocity of a rigid body $\boldsymbol{\omega}$ and its moments of inertia. The angular momentum of a rigid body is defined with respect to a reference frame $\mathcal{B}$ attached to the rigid body and moving with it. By choosing the body frame coincident with the center of mass of the system, the angular momentum can be calculated as:

$$\mathbf{h}^{\mathcal{B}} = \mathbf{J}\boldsymbol{\omega} \tag{2.44}$$

where $\mathbf{J}$ is a 3×3 symmetric matrix representing the vehicle's inertia tensor and $\boldsymbol{\omega}$ is the angular velocity of the vehicle with respect to the inertial frame $\mathcal{I}$ represented in the body frame $\mathcal{B}$.

The fundamental equation for rigid body motion in inertial frame $\mathcal{I}$ is:

$$\frac{d\mathbf{h}^{\mathcal{I}}}{dt} = \boldsymbol{\tau}^{\mathcal{I}} \tag{2.45}$$

where $\boldsymbol{\tau}^{\mathcal{I}}$ are the external torques expressed in the inertial frame. A more convenient way is to derive the dynamical equations as observed in the body frame $\mathcal{B}$. By recalling Eq. 2.38, it is possible to write:

$$\frac{d}{dt}\mathbf{h}^{\mathcal{B}} = \dot{\mathbf{h}}^{\mathcal{B}} = \boldsymbol{\tau}^{\mathcal{B}} - S(\boldsymbol{\omega})(\mathbf{J}\boldsymbol{\omega}) \tag{2.46}$$

which is known as Euler equation and relates the angular momentum to the external torques applied to the body.

For constant inertia $\mathbf{J}$, the angular velocity $\boldsymbol{\omega}$ is directly related to the external torques as:

$$\mathbf{J}\dot{\boldsymbol{\omega}} = -S(\boldsymbol{\omega})\boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\tau}. \tag{2.47}$$

2.4 Controllers

In this section, the some of the most common controller used for UAVs are presented, namely the PID, the Linear Quadratic Regulator (LQR), and the MPC.

2.4.1 Proportional-Derivative-Integral

The PID is one of the most common in the UAV field since it is intuitive to tune and only requires limited knowledge of the system. The general form of PID controller is:

$$u_{PID}(t) = K_P\, e(t) + K_I \int_0^t e(\tau)d\tau + K_D\, \frac{de(t)}{dt} \tag{2.48}$$

where $e(t) = x_{ref}(t) - x(t)$ represents the error vector and K_P, K_I, and K_D correspond to the proportional, integral, and derivative gains, respectively. Some disadvantages of this kind of controller is that it is designed considering Single Input and Single Output linear systems and does not respect the limits of the actuators.

2.4.2 Linear Quadratic Regulator

LQR is an optimal control technique applied to linear systems of the form $\dot{\mathbf{x}} = \mathbf{A}\,\mathbf{x} + \mathbf{B}\,\mathbf{u}$. It aims to find the optimal gain matrix ($\mathbf{K}$) for the linear control law

$$\mathbf{u}_{LQR} = -\mathbf{K}\,\mathbf{x}. \tag{2.49}$$

The LQR is said to be optimal because it minimizes the cost function J. For a horizon T, it is defined as:

$$J = \int_0^T \left[\mathbf{x}(\tau)^T\,\mathbf{Q}\,\mathbf{x}(\tau) + \mathbf{u}(\tau)^T\,\mathbf{R}\,\mathbf{u}(\tau)\right] d\tau \tag{2.50}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are symmetric matrices, with $\mathbf{Q}$ positive semi-definite and $\mathbf{R}$ positive definite. $\mathbf{Q}$ and $\mathbf{R}$ are called state weighting matrix and control weighting matrix respectively. In the cost function, the quadratic form $\mathbf{x}(\tau)^T\,\mathbf{Q}\,\mathbf{x}(\tau)$ represents a penalty on the deviation of the state $\mathbf{x}$ from the origin, while $\mathbf{u}(\tau)^T\,\mathbf{R}\,\mathbf{u}(\tau)$ represents the cost of control.

By fixing the value of T, the control interval is limited, which means that the control law shall minimize the cost function in finite time. On the other hand, if the terminal time is defined as infinite, it means that the interest is in the behavior of the system starting from the current time up to the steady state. For the infinitive horizon problem, it can be shown [17] that the solution which minimizes the cost function and guarantees closed-loop asymptotic stability is the constant state feedback law:

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P}, \tag{2.51}$$

where $\mathbf{P}$ is the solution of the Algebraic Riccati Equation (ARE):

$$\mathbf{P}\,\mathbf{A} + \mathbf{A}^T\,\mathbf{P} - \mathbf{P}\,\mathbf{B}\,\mathbf{R}^{-1}\mathbf{B}^T\,\mathbf{P} + \mathbf{Q} = \mathbf{0}. \tag{2.52}$$

For the ARE to have a unique, positive definite solution $\mathbf{P}$ that minimizes the cost function when the defined control law is applied, it is sufficient that the system defined by the pair $(\mathbf{A}, \mathbf{B})$ is controllable.

2.4.3 Model Predictive Control

MPC is an important advanced control technique for difficult multivariable control problems, which makes use of a process model to predict the future behavior of the controlled system. This subsection aims to give the key concepts of MPC and is based on [18, 19].

MPC explicitly use a model to predict the future behavior of the system. Based on current measurements and predictions of the future values of the outputs and states, the objective of MPC is to determine a sequence of control actions such that the predicted response of system the moves to the reference in an optimal manner in a finite-horizon approach. Since MPC is mostly employed on digital processors, only the discrete-time formulations are covered here.

The strategy employed by MPC can be explained as follows: at each time sampling instant k, the state of the plant is sampled and an optimization problem is numerically solved for a finite time horizon T_p in the future, called the prediction horizon, and determines over a control horizon T_c (where $T_c \leq T_p$) the control actions $\mathbf{u}$ such that an objective function is optimized. The control signal is assumed to be piecewise constant on each of the N_c predicted sampling times, with a regular sampling interval. Assuming that the output of the optimization is the control sequence $\mathbf{U}^*$, only the first control input is applied to the real system. Then, another sample is taken and the process is repeated using the new state of the system, yielding a new control and new predicted state path. Since at each instant the horizon is displaced towards the future, MPC is also known as receding horizon control or moving horizon control. Figure 2.2 exemplifies the general concept of MPC.

Given an arbitrary system of the form $\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k)$, MPC minimizes a user-defined cost function J, which is usually in the standard quadratic form. The optimization problem solved at each time instant can thus be formulated as:

$$\min_{\bar{\mathbf{u}}_k,\cdots,\bar{\mathbf{u}}_{k+N-1}} \quad ||\bar{\mathbf{x}}_{k+N} - \mathbf{x}^{\text{ref}}_{k+N}||^2_{\mathbf{R}} + \sum_{i=0}^{N-1} \left(||\bar{\mathbf{x}}_{k+i} - \mathbf{x}^{\text{ref}}_{k+i}||^2_{\mathbf{Q}} + ||\bar{\mathbf{u}}_{k+i} - \mathbf{u}^{\text{ref}}_{k+i}||^2_{\mathbf{P}} \right) \tag{2.53a}$$

$$\text{subject to} \quad \bar{\mathbf{x}}_{k+i+1} = f(\bar{\mathbf{x}}_{k+i}, \bar{\mathbf{u}}_{k+1}),\ i = 0, \cdots, N-1 \tag{2.53b}$$

$$\bar{\mathbf{x}}_k = \mathbf{x}_k \tag{2.53c}$$

$$\bar{\mathbf{x}}_{k+i} \in \mathfrak{X},\ i = 0, \cdots, N-1 \tag{2.53d}$$

$$\bar{\mathbf{x}}_{k+N} \in \mathfrak{X} \tag{2.53e}$$

$$\bar{\mathbf{u}}_{k+1} \in \mathfrak{U},\ i = 0, \cdots, N-1 \tag{2.53f}$$

where k is the current sample time; i is index of the prediction horizon; N is the prediction horizon length; $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ are, respectively, the weight matrices on the control action error, state error, and final state error; $\bar{\mathbf{x}}_{k+i}$ is the state vector of the predicted system,

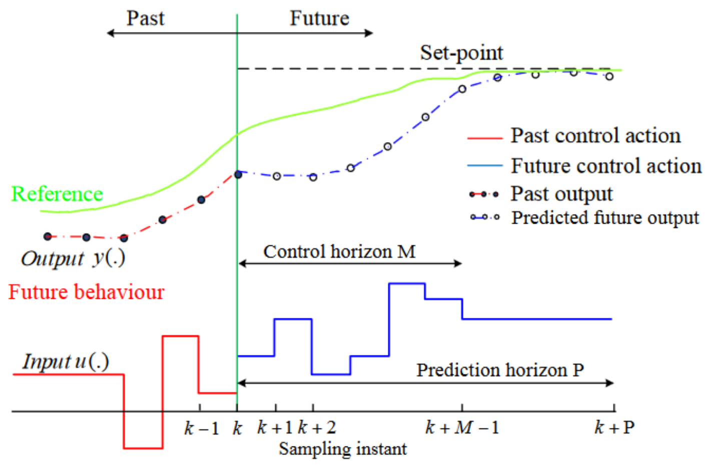


Figure 2.2: General Concept of Model Predictive Control for an arbitrary system and set-point. Extracted from [20].

$\mathbf{x}_{k+i}^{\text{ref}}$ is the state reference signal; $\overline{\mathbf{u}}_{k+i}$ is the predicted optimal control action; $\mathbf{u}_{k+i}^{\text{ref}}$ is the control reference, the subscript $k+i$ denotes the sample of a time signal at i steps ahead of the current time k, and $k+i+1$ indicates the next evolution of that step. Finally, $\mathfrak{U}$ and $\mathfrak{X}$ represent the set of input and state constraints respectively and $\mathbf{x}_k$ is the sample (or estimate) of the state vector at the current iteration at instant k.

Notice that the equations of motion are given in discrete time. However, usually it is more convenient to derive the dynamics of the system in continuous time. This is handle by employing a suitable discretization technique, such as Euler and Runge-Kutta methods. Since the dynamics equality constraints in this dissertation are mostly nonlinear, the control strategy becomes NMPC.

The system model used to predict the future in the controller is initialized by the actual system (Eq. 2.53c). Thus, the state is assumed to be measured or must be estimated. The distinction between the real system and the variables in the controller is necessary because, even in the nominal undisturbed case, the predicted values in general will not be the same as the actual closed-loop values, since the optimal input is recalculated at every sampling time [19].

The matrices $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$ are real and symmetric. Particularly, $\mathbf{Q}$ should be positive semi-definite, while $\mathbf{P}$ and $\mathbf{R}$ should be positive definite. These matrices can be tuned to obtain the desired optimal solution. The solution of the optimization problem is an optimal control sequence $\mathbf{U}^*$ of the form:

$$\mathbf{U}^* = [\mathbf{u}_k^*, \mathbf{u}_{k+1}^*, \ldots, \mathbf{u}_{k+N-1}^*] \tag{2.54}$$

where only the first term $\mathbf{u}_k^*$ is applied to the plant.

As mentioned in [18], some key aspects to aspects to consider in the MPC are the model, the cost function, the constraints, the choice of solver, and other parameters of the controller.

The model is the essence of MPC. As pointed out by [21], "models are not perfect forecasters, and feedback can overcome some effects of poor models, but starting with a poor process model is akin to driving a car at night without headlights; the feedback may be a bit late to be truly effective". The nonlinear optimization requires larger computation, thus the prediction model should aim to be highly descriptive while keeping the number of states and inputs minimal to avoid excessive computational overhead. The choice of initial values for the optimizer significantly impacts the course it follows. By making use of the last solution as a warm start, the computational time can significantly reduce. The key idea is that the solution should vary minimally between consecutive control intervals.

The constraints used in the optimization formulation generally affect the time required to solve the problem and affects its feasibility. Hard constraints on the inputs $\mathbf{u}$ represent physical limitations, which must not be violated. On the other hand, hard constraints on the state $\mathbf{x}$ are often more desired than required. One possible way to overcome them is to introduce slack variables to the optimization problem, creating an extra degree of freedom. Then, the objective function is modified such that the extent of the violation is penalized in the cost function.

The prediction horizon N is the number of future control intervals the MPC must evaluate by prediction. Ideally one would like to use an infinite horizon in NMPC formulation, since in the nominal case, the closed-loop trajectories coincide with the open-loop predicted ones [19]. However, infinite horizon schemes generally cannot be implemented in practice, since the open-loop optimal control problem cannot be solved. Nevertheless, using finite horizon, the repeated minimization over a finite horizon objective in a receding horizon manner does not lead to an optimal solution for the infinite horizon problem. From this, there is a trade-off between a desirable horizon from a computational point of view and the performance obtained by using longer horizons. The prediction horizon must be long enough to capture the effect of a change of the manipulated variable $\mathbf{u}$.

The control horizon N_c is the number of values of the input $\mathbf{u}$ that a MPC controller calculates at each time instant. It can assume values between 1 and N and it is used to reduce the number of optimized input variables.

Chapter 3

State of the Art

This chapter includes the relevant work review on recovery control of UAVs, controllers used for tailsitters and deep stall control.

3.1 Recovery Control

The topic of recovery control is mostly applied for rotorcrafts. For example, Faessler et. al. [22] and [23] recovery procedures for quadrotors. The attitude of the vehicle is represented using unit quaternions and the recovery procedure is divided into five sequential steps:

1. Attitude Control: the quadrotor control its orientation to be horizontal.
2. Attitude and Velocity Control: control the vertical velocity to zero.
3. Attitude and Height Control: stabilize the height relative to the ground together with the attitude.
4. Horizontal Velocity Control: decrease the horizontal velocity before locking the position.
5. Position Control: maintain hovering over a desired position.

For all recovery stages, a high level Proportional Derivative (PD) controller that tracks the position and velocity was employed with different gains based on the recovery step. The attitude controller is based on quaternions and the attitude control is separated into two parts, one dealing with the inclination error and the other with the heading.

MPC leverages the model of the system to calculate optimal control inputs by solving an optimization problem in real time. This approach allows it to account for the UAV's dynamics and constraints, including actuator limits, making it highly effective for managing the recovery process. Studies like [24, 25] demonstrate the use of this approach to recover rotorcrafts by exploiting the vehicle's dynamics. In scenarios involving rotor failure in a quadrotor, [26] addresses the full nonlinear dynamics of a damaged UAV. The cost function in their formulation separates the attitude error into heading and inclination rotations, acknowledging that yaw control becomes unachievable when one rotor is non-functional.

The topic of recovery control of tailsitter is rather limited. In [10], a full flight controller was designed by dividing the flight envelope of the aircraft into four different modes: hover, transition, level, and recovery. The latter is triggered whenever the vehicle state reaches unexpected values and drives the aircraft towards stable hovered flight. The controller was designed based on back-stepping procedure by assuming that the motion of the vehicle is restricted to a vertical plane. The attitude is represented by the non-standard parameterization Z-X-Y of Euler angles. The saturation of actuators was neglected and only numerical simulations were used to assess the performance of the controller.

In [27], a global controller is developed for flying wing tailsitter vehicle. A cascaded control strategy is used, with an inner loop that tracks the desired attitude using a lookup table with precomputed optimal attitude trajectories, such that the reorientation maneuver minimizes the torque requirement around the weakest axes. The controller was validated through extensive experimentation, in which the vehicle was able to recover from large attitude errors.

3.2 Tailsitter Attitude Control

The topic of attitude control for tailsitter is mostly focused on the transition maneuver and the aggressive flight, for example [28]. For instance, in [28], a global control formulation is proposed, enabling agile maneuvers in the whole flight envelope, including uncoordinated flight with sideslip. The dual rotor tailsitter with two flaps is shown to be differentially flat, which means that the inputs and states can be expressed as a function of the flat output and a finite number of its derivatives. The controller is able to track reference position, acceleration, and jerk (the third derivative of the position), as well as yaw angle and yaw rate. Several components, based on various methodologies, such as PID and nonlinear dynamic inversion, were adopted in the controller designed. A simplified aerodynamic model based on ϕ theory [29] was used, allowing for a nine scalar coefficient aerodynamic representation. Unmodeled forces were estimated by comparing the measured acceleration to the expected acceleration according to the vehicle aerodynamics model and motor speed measurements. Particularly, the attitude is controlled by a PD controller and the attitude is represented using quaternions. The proposition was validated in real life in indoor experiments.

The problem of attitude control in a tailsitter can be effectively addressed by decomposing the attitude error into two distinct components: inclination and heading. As shown in [30, 7], this strategy overcomes the issues with standard quaternion feedback in the presence of large attitude errors and can mitigate limitations in control moments, particularly during hovering.

Regarding a quadrotor tailsitter UAV without control surfaces such as the Swan-K1, which is the focus of this dissertation, some relevant work is seen in [31, 11, 32, 33, 34]. Full wind tunnel measurements were performed to model the aerodynamics of the vehicle and different types of controllers were used, such as PID [31] and MPC [32, 33]. An unified controller that is able to uniformly treat the hovering and forward flight, enabling continuous transitions between the flight modes, is developed in [11]. In [34], an optimization-based trajectory planner is proposed to design high-quality, smooth trajectories with consideration of kinodynamics constraints, singularity-free constraints, and actuator saturation.

3.3 Deep stall control of UAV

Deep stall is a flight condition in which the AoA is very high, beyond the stall angle, and usually causes the aircraft to lose speed and altitude. When the AoA increases, the boundary layer of the airflow separates from the airfoil of the wing, creating a turbulence wake behind the wing, and causing a massive reduction in the lift and an increase of drag. The AoA when it occurs is called the stall angle and deep stall refers to AoAs higher than the stall angle from which it is difficult to recover.

To enable deep stall landing of a fixed-wing UAV, MPC was used in [35, 36]. The aerodynamical model only takes into account the longitudinal dynamics, such that the lift and drag coefficients only depend on the AoA and only the torque about the body's y axis is needed. To represent the effects of aerodynamics on the predictive model, a roughly fitted polynomial curve was used to calculate the drag coefficient, whereas the lift coefficient is calculated by blending the linear part, in which the angle of is small, and the post stall part, modeled as a flat-plate. By using this model, NMPC is used to land the UAV at a given location, a given path angle, and at a minimal speed. The lift of the vehicle is decreased until it is unable to keep the UAV leveled while the drag increases, reducing the vehicle's speed. Only simulations were shown to validate the proposed methodology.

Chapter 4

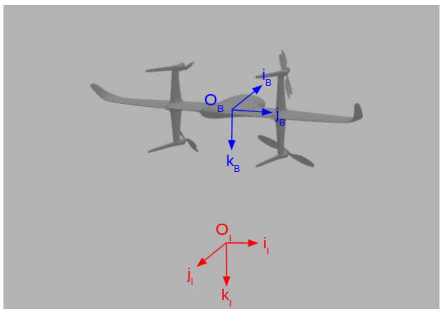
System Modeling

This chapter presents the mathematical model of the quadrotor tailsitter without control surfaces used in this dissertation and was based on [34, 5]. It additionally presents the simulation setup.

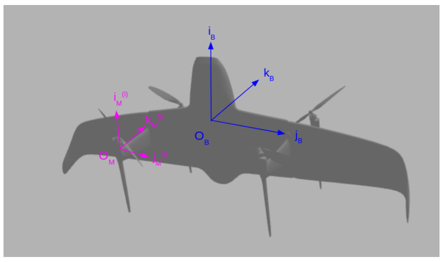
4.1 Reference Frames

The definition of coordinate frames follows the convention of traditional fixed-wing aircraft. The Inertial Reference Frame $\mathcal{I}$ $\{\mathbf{O}_\mathcal{I}, \hat{\mathbf{i}}_\mathcal{I}, \hat{\mathbf{j}}_\mathcal{I}, \hat{\mathbf{k}}_\mathcal{I}\}$ is defined according to the North-East-Down (NED) convention. It is fixed at some point $\mathbf{O}_\mathcal{I}$ in the Earth's surface, which is considered flat and still, and is identified by the set of perpendicular unitary vectors $\{\hat{\mathbf{i}}_\mathcal{I}$, $\hat{\mathbf{j}}_\mathcal{I}$, $\hat{\mathbf{k}}_\mathcal{I}\}$, in which $\hat{\mathbf{i}}_\mathcal{I}$ points to the geographic North and is parallel to the ground, $\hat{\mathbf{j}}_\mathcal{I}$ points to the East and is also parallel to the ground, and the $\hat{\mathbf{k}}_\mathcal{I}$ is directed to the center of the Earth, completing the right-handed set. The Body Reference Frame $\mathcal{B}$ $\{\mathbf{O}_\mathcal{B}, \hat{\mathbf{i}}_\mathcal{B}, \hat{\mathbf{j}}_\mathcal{B}, \hat{\mathbf{k}}_\mathcal{B}\}$ is defined by the set of perpendicular unitary vectors $\{\hat{\mathbf{i}}_\mathcal{B}$, $\hat{\mathbf{j}}_\mathcal{B}$, $\hat{\mathbf{k}}_\mathcal{B}\}$, where $\mathbf{O}_\mathcal{B}$ is the vehicle's center of gravity and $\hat{\mathbf{i}}_\mathcal{B}$ is directed towards the aircraft nose, $\hat{\mathbf{j}}_\mathcal{B}$ towards the right wing, and $\hat{\mathbf{k}}_\mathcal{B}$ points down, completing the right handed set. These reference frames are seen in Fig. 4.1a.

Additionally, since the rotors are slightly tilted, a motor frame $\mathcal{F}_\mathcal{M}^{(i)}$ $\{\mathbf{O}_\mathcal{M}^{(\mathbf{i})}, \hat{\mathbf{i}}_\mathcal{M}^{(\mathbf{i})}, \hat{\mathbf{j}}_\mathcal{M}^{(\mathbf{i})}, \hat{\mathbf{k}}_\mathcal{M}^{(\mathbf{i})}\}$ is defined for each motor i, with $i = 1, 2, 3, 4$. The $\hat{\mathbf{i}}_\mathcal{M}^{(i)}$ axis of the motor frame is aligned with the i-th motor rotation axis and stays still with the motor stator, as seen in Fig. 4.1b. The orientation of the motor frame is fixed with respect to the body frame $\mathcal{B}$ and is denoted as $\mathbf{R}_\mathcal{M}^{(i)}$.



(a) Body frame $\mathcal{B}$ in blue and inertial frame $\mathcal{I}$ in red.



(b) Body frame $\mathcal{B}$ in blue and one of the motors frame $\mathcal{F}_{\mathcal{M}}^{(i)}$ in pink.

Figure 4.1: Definition of coordinate frames for quadrotor tailsitter.

4.2 Airframe Dynamics

Newton's second law can be used to derive the equations of motion of the vehicle, which is modelled as a rigid-body.

Let the vector $\mathbf{p}^{\mathcal{I}} = \begin{bmatrix} x, & y, & z \end{bmatrix}^T$ denote the position of the center of mass of the UAV expressed in the inertial frame $\mathcal{I}$. Also, let the orientation of the aircraft's body-fixed frame $\mathcal{B}$ with respect to the inertial frame $\mathcal{I}$ be represented by a rotation matrix ${}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R} \in \mathbb{R}^{3\times3}$. Let $\mathbf{v}^{\mathcal{I}} = \begin{bmatrix} u, & v, & w \end{bmatrix}^T$ be the velocity of the center of mass relative to the fixed frame in the inertial frame and $\boldsymbol{\omega}^{\mathcal{B}}_{\mathcal{I}\mathcal{B}} = \begin{bmatrix} p, & q, & r \end{bmatrix}^T$ represent the angular velocity of $\mathcal{B}$ relative to $\mathcal{I}$ in the body frame.

In order to simplify the notation, in the remainder of the thesis, the following conventions will be adopted: $\mathbf{p} = \mathbf{p}^{\mathcal{I}}$, $\mathbf{R} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}$, $\mathbf{v} = \mathbf{v}^{\mathcal{I}}$, $\boldsymbol{\omega} = \boldsymbol{\omega}^{\mathcal{B}}_{\mathcal{I}\mathcal{B}}$.

Therefore, as seen in [34], the dynamics of the hybrid UAV for the center of mass is given by:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{4.1}$$

$$\dot{\mathbf{v}} = g\,\mathbf{e_3} + \frac{1}{m}\mathbf{R}(\mathbf{F}_r + \mathbf{F}_a) \tag{4.2}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -S(\boldsymbol{\omega}) \end{bmatrix}\boldsymbol{q} \tag{4.3}$$

$$\mathbf{J}\,\dot{\boldsymbol{\omega}} = -S(\boldsymbol{\omega})\,\mathbf{J}\,\boldsymbol{\omega} + \boldsymbol{\tau} + \mathbf{M}_a \tag{4.4}$$

where $\mathbf{e_3} = \begin{bmatrix} 0, & 0, & 1 \end{bmatrix}^T$ and g is the gravitational acceleration. Furthermore, $\mathbf{J}$ is the inertia tensor of the vehicle and m is the mass. $\mathbf{F}_a$ and $\mathbf{M}_a$ denote the forces and moments applied to the vehicle in the body frame due to the aerodynamic effects represented by the subscript a. Finally, $\mathbf{F}_r$ and $\boldsymbol{\tau}$ denote the force and torque applied due to the rotors. Notice that the forces $\mathbf{F}_a$ and $\mathbf{F}_r$ as well as the moments $\mathbf{M}_a$ and $\boldsymbol{\tau}$ are all calculated with respect to the body frame $\mathcal{B}$. The motor dynamics are considered significantly faster than the UAV body dynamics and are thus neglected. Notice that since both controllers designed employ unit quaternions, the attitude is represented by the quaternion $\boldsymbol{q}$. Then, the rotation matrix $\mathbf{R}$ in Eq. 4.2 can be calculated by Eq. 2.33.

4.3 Forces and Moments

As a general case, the forces that act on a tailsitter UAV can be divided into three main categories: gravitational, the generated by the propellers, and aerodynamic.

4.3.1 Gravitational force

The gravitational force always acts on the center of gravity of the aircraft and its direction is parallel to $\hat{\mathbf{k}}_{\mathcal{I}}$. For UAVs the gravity is assumed to be constant, an assumption valid as long as the UAV operates close to the Earth's surface, where variations in gravitational acceleration are negligible.

4.3.2 Propeller forces and moments

The force generated by the propellers is in the same direction of the propellers, which are fixed relatively to the body frame. In the case of the Swan-K1, the propellers are slightly tilted (which can increase the rotation moment around the $\hat{\mathbf{i}}_{\mathcal{B}}$ axis [5]). Therefore, it is necessary to consider the tilted direction of each propeller to compute the combined propeller force.

The modeling of propeller generated forces $\mathbf{F}_r \in \mathbb{R}^3$ and torques $\mathbf{M}_r \in \mathbb{R}^3$ can include several different effects, for instance gyroscopic torques and free-stream velocity over the propeller blades [37]. However, in most applications, only the most significant terms are considered for designing the controller, specifically the thrust and resisting torque of each propeller, to simplify the model.

The thrust T_i generated by the i-th motor has the same direction of $\hat{\mathbf{i}}_{\mathcal{M}}^{(i)}$, such that the total force $\mathbf{F}_r$ generated by the propellers in the body frame is given by:

$$\mathbf{F}_r = \sum_{i=1}^{4} \mathbf{R}_{\mathcal{M}}^{(i)} \begin{bmatrix} T_i \\ 0 \\ 0 \end{bmatrix}. \tag{4.5}$$

The moment $\boldsymbol{\tau}$ generated by the propellers is given by:

$$\boldsymbol{\tau} = \sum_{i=1}^{4} \left(S(\mathbf{r}_i)\mathbf{R}_{M}^{(i)} \begin{bmatrix} T_i \\ 0 \\ 0 \end{bmatrix} + \mathbf{R}_{M}^{(i)} \begin{bmatrix} -s_i Q_i \\ 0 \\ 0 \end{bmatrix} \right) \tag{4.6}$$

where $\mathbf{r}_i$ is the position vector of each motor with respect to the body frame, s_i indicates the rotation direction of the i-th propeller, and Q_i is the moment produced by the i-th propeller. Note that adjacent propellers rotate in opposite directions; if a propeller is spinning clockwise, then the two adjacent ones will be rotating counter-clockwise, so that torques are balanced if all propellers rotate at the same rate.

The thrust T_i and the torque Q_i produced by the i-th rotor are given by:

$$T_i = c_t \, \omega_i^2 \tag{4.7}$$

$$Q_i = c_q \, \omega_i^2 = \kappa T_i \tag{4.8}$$

in which c_t, c_q are, respectively, the thrust and torque coefficients, κ is the ratio between them, and ω_i denotes the rotational speed of the i-th rotor. These coefficients depend on the characteristics of the propellers and can be found by load cell experiments.

4.3.3 Aerodynamic forces and moments

The aerodynamic forces are in general generated by the aerodynamic surfaces of the vehicle: wings, fuselage, tail, and, if present, control surfaces. When contrasted with traditional fixed-wing or rotary aircraft, the dynamic modeling of hybrid vehicles becomes more intricate. This complexity arises due to the need for comprehensive consideration of the diverse aerodynamic characteristics inherent in these components across various flight conditions or operational envelopes. Another difficulty that comes with modeling a hybrid aircraft is to understand aerodynamical interference between the propeller and the wing. The modeling of the aerodynamics of the aircraft can be derived from Wind tunnel measurements [7], Computational Fluid Dynamics (CFD) analysis [38], or real flight experiment data [39].

In [5], wind tunnel measurements are used to model the aerodynamic force $\mathbf{F}_a$ and moment $\mathbf{M}_a$, which are functions of the speed of the vehicle relative to its surrounding air mass, that is, the airspeed. Considering that the speed of the vehicle relative to the inertial frame is $\mathbf{v}$ and that the wind speed relative to the earth is $\mathbf{w}$, the airspeed $\mathfrak{u}$ is calculated by:

$$\mathfrak{u} = \mathbf{R}^T(\mathbf{v} - \mathbf{w}) \tag{4.9}$$

where $\mathfrak{u}$ is given in the body frame $\mathcal{B}$ and the velocity of the vehicle $\mathbf{v}$ and the wind $\mathbf{w}$ are given in the inertial frame $\mathcal{I}$.

With $\mathfrak{u}$ defined, it is possible to calculate the AoA, denoted by α, and sideslip angle β:

$$V = \sqrt{\mathfrak{u}_x^2 + \mathfrak{u}_y^2 + \mathfrak{u}_z^2} \tag{4.10}$$

$$\alpha = \arctan 2\left(\frac{\mathfrak{u}_z}{\mathfrak{u}_x}\right) \tag{4.11}$$

$$\beta = \arcsin\left(\frac{\mathfrak{u}_z}{V}\right) \tag{4.12}$$

where $\mathfrak{u}_x$, $\mathfrak{u}_y$, and $\mathfrak{u}_z$ are the components of the airspeed $\mathfrak{u}$ in the $\hat{\mathbf{i}}_\mathcal{B}$, $\hat{\mathbf{j}}_\mathcal{B}$, and $\hat{\mathbf{k}}_\mathcal{B}$ direction, respectively. When the airspeed $\mathfrak{u}$ is zero, there is no angle of attack and no side-slip angle.

The aerodynamic force $\mathbf{F}_a$ is usually characterized by Lift force $\boldsymbol{\mathcal{L}}$, Drag force $\boldsymbol{\mathcal{D}}$, and Side Force $\boldsymbol{\mathcal{Y}}$. While the latter is along the body y axis, the Lift and Drag forces are respectively along and perpendicular to the airspeed projection onto the body $x - z$ plane. The resulting aerodynamic force with respect to the body frame $\mathcal{B}$ is calculated by:

$$\mathbf{F}_a = \begin{bmatrix} -\cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & -\cos\alpha \end{bmatrix} \begin{bmatrix} \boldsymbol{\mathcal{D}} \\ \boldsymbol{\mathcal{Y}} \\ \boldsymbol{\mathcal{L}} \end{bmatrix}. \tag{4.13}$$

The aerodynamic moment can be decomposed into rolling $\mathbf{L}$, pitching $\mathbf{M}$, and yawing $\mathbf{N}$ moments:

$$\mathbf{M}_a = \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{N} \end{bmatrix}. \tag{4.14}$$

The aerodynamic forces and moments can be further parameterized as:

$$\boldsymbol{\mathcal{L}} = \frac{1}{2}\rho V^2 S C_L \tag{4.15}$$

$$\boldsymbol{\mathcal{D}} = \frac{1}{2}\rho V^2 S C_D \tag{4.16}$$

$$\boldsymbol{\mathcal{Y}} = \frac{1}{2}\rho V^2 S C_Y \tag{4.17}$$

$$\mathbf{L} = \frac{1}{2}\rho V^2 S \bar{c} C_l \tag{4.18}$$

$$\mathbf{M} = \frac{1}{2}\rho V^2 S \bar{c} C_m \tag{4.19}$$

$$\mathbf{N} = \frac{1}{2}\rho V^2 S \bar{c} C_n \tag{4.20}$$

where S is the reference area, which is usually the wing area; $\bar{c}$ is the characteristic length, which is usually the mean aerodynamic chord. V is the magnitude of the airspeed, as seen in Eq. 4.10; C_L, C_D, and C_Y are respectively the lift, drag, and side force coefficients. C_l, C_m, C_n are, respectively the rolling, pitching, and yawing moments coefficients. All these coefficients are generally dependent on α (Eq. 4.11), the sideslip angle β (Eq. 4.12), and the airspeed magnitude V (Eq. 4.10), as seen in [5].

4.4 Simulation Setup

This section presents the relevant aspects of the simulation environment which was used to test the two methodologies proposed.

The simulation environment is the Gazebo, an open-source 3D dynamic simulator for robots [40]. It can simulate the dynamics of the vehicle, the sensors used, and include different conditions for the environment, such as wind. The PX4 SITL is compatible with Gazebo and allows developers to test and validate the behavior of UAVs using the PX4 autopilot software, without requiring actual hardware. The UAV control, state estimation, and data communication are performed by modules that are incorporated in the PX4 Autopilot. This firmware is compatible with several versions of PixHawk which is an independent open-hardware project providing readily-available, low-cost, and high-end, autopilot hardware designs to the academic, hobby and industrial communities [1]. Therefore, the PX4 SITL allows for a relatively easy transition from a simulation environment to real world experiments as it simulates the flight controller.

[1] `https://docs.px4.io/main/en/flight_controller/pixhawk_series`. Accessed: October 2024.

The controllers developed in this research are meant to run offboard the PixHawk, by providing thrust and torques setpoints via ROS2 (Robot Operating System 2), which is an open source software development kit for robotics applications [41]. The ROS2-PX4 architecture provides a deep integration between ROS2 and PX4, allowing ROS2 subscribers or publisher nodes to interface directly with PX4 uORB topics [2].

4.4.1 Simulation model of quadrotor tailsitter

The model of a quadrotor tailsitter is provided in the PX4-repository [3]. In Gazebo, the description of robots, objects, and the environment is done using the Simulation Description Format (SDF), which essentially uses a collection of links, joints, collision objects, visuals, and plugins, allowing for multi-body parts to be described separately. By analyzing the file, it is possible to extract the parameters that describe the UAV, such as the mass m, moment of inertia $\mathbf{J}$, and geometry, as well as the parameters of the simulated environment, including the gravitational acceleration g and air density ρ.

The rotors are modeled using the *gazebo_motor_model* plugin, which takes as input the rotor motor constant and the moment coefficient to calculate the thrust and the moment produced by each rotor, as in Eqs. 4.7-4.8. To enhance the realism of the simulation, additional effects are included, such as PID controllers that regulate rotor velocities and the modeling of the advance ratio, which captures the relationship between rotor speed and forward motion. These factors contribute to a more accurate representation of the rotor dynamics in various flight conditions.

The main parameters used in the simulation are shown in Table 4.1.

[2] `http://docs.px4.io/main/en/ros2/user_guide`. Accessed: August 2024.

[3] `https://github.com/PX4/PX4-SITL_gazebo-classic/tree/main/models/quadtailsitter`. Accessed: August 2024.

Parameter	Value	Description
g	9.81 m s^{-2}	Earth's gravity
ρ	1.2041 kg m^{-3}	Air density
m	1.635 kg	Mass of UAV
J_x	0.113 kg m	x-axis moment of inertia
J_y	0.302 kg m	y-axis moment of inertia
J_z	0.8354 kg m	z-axis moment of inertia
ω_{max}	1200 rad s^{-1}	Maximum motor speed
c_t	$8.54858 \cdot 10^{-6}$ Ns^2/rad^2	Motor thrust coefficient
κ	0.06 m	Moment constant

Table 4.1: Physical parameters used in simulation.

One of the most important aspects when simulating an UAV with wings is the aerodynamic modeling. In the context of Gazebo, there are two mainly used plugins for the aerodynamics of the vehicle: the *lift_and_drag_plugin* [4], and the *advanced_lift_and_drag_plugin* [5]. On one hand, the former computes the lift and drag coefficients based solely on the AoA of the vehicle. The alpha/lift coefficient and alpha/drag coefficient curves are simplified as two lines each, instead of a smooth curve, as shown in Fig. 4.2. With the coefficients computed, the lift and drag forces are calculated as shown in Eqs. 4.15 and 4.16, respectively.

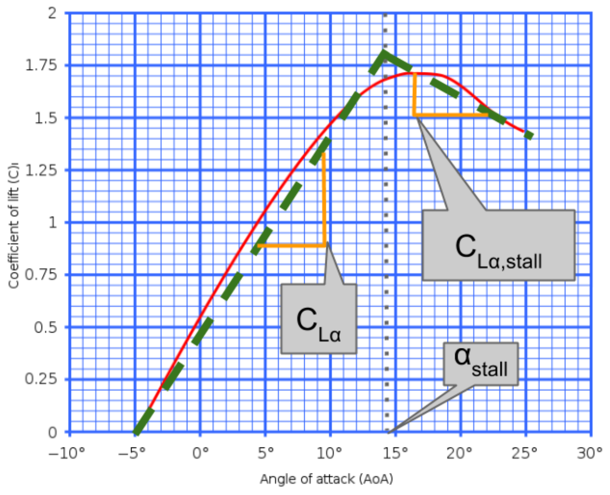


Figure 4.2: Lift curve computed by lift_and_drag_plugin. Extracted from [5]. The values used to generate the curve were arbitrarily selected.

[4] `https://classic.gazebosim.org/tutorials?tut=aerodynamics`. Accessed: 10-05-2024

[5] `https://gazebosim.org/api/sim/8/classgz_1_1sim_1_1systems_1_1AdvancedLiftDrag.html`. Accessed: 10-05-2024

As it is clear from Fig. 4.2, the plugin does include nonlinearities expected at a high angle of attack.

On the other hand, the *advanced_lift_and_drag_plugin* includes a quadratic formulation for drag, computes side force, models post-stall behavior as a flat-plate, and includes aerodynamic moments about all three axes. To model the aerodynamics, the plugin receives parameters from Athena Lattice Vortex to calculate the coefficients which are then used to described the aerodynamics. The aerodynamic coefficients C_L, C_D, C_Y, C_l, C_m, and C_n (Eqs. 4.15-4.20) are calculated by considering different effects during the flight. For the linear, pre-stall flight, the following equations are used:

$$C_L = C_{L0} + C_{L\alpha}\alpha + C_{L\beta}\beta + C_{Lp}p + C_{Lq}q + C_{Lr}r + \sum_{x=1}^{CS} C_{L,ctrl,x}\delta_{ctrl,x} \tag{4.21}$$

$$C_D = C_{D0} + \frac{{C_L}^2}{\pi AR\,e} + C_{D\beta}\beta + C_{Dp}p + C_{Dq}q + C_{Dr}r + \sum_{x=1}^{CS} C_{D,ctrl,x}\delta_{ctrl,x} \tag{4.22}$$

$$C_Y = C_{Y\alpha}\alpha + C_{Y\beta}\beta + C_{Yp}p + C_{Yq}q + C_{Yr}r + \sum_{x=1}^{CS} C_{Y,ctrl,x}\delta_{ctrl,x} \tag{4.23}$$

$$C_l = C_{l\alpha}\alpha + C_{l\beta}\beta + C_{lp}p + C_{lq}q + C_{lr}r + \sum_{x=1}^{CS} C_{l,ctrl,x}\delta_{ctrl,x} \tag{4.24}$$

$$C_m = C_{m0} + C_{m\alpha}\alpha + C_{m\beta}\beta + C_{mp}p + C_{mq}q + C_{mr}r + \sum_{x=1}^{CS} C_{m,ctrl,x}\delta_{ctrl,x} \tag{4.25}$$

$$C_n = C_{n\alpha}\alpha + C_{n\beta}\beta + C_{np}p + C_{nq}q + C_{nr}r + \sum_{x=1}^{CS} C_{n,ctrl,x}\delta_{ctrl,x} \tag{4.26}$$

where the subscripts of each aerodynamic coefficient denote the variable with respect to which the coefficient is evaluated. Specifically, the notation 0 signifies the value of the coefficient at zero angle of attack (AoA) for the lift and pitching moment or zero lift for the drag. In contrast, α represents the angle of attack; β indicates the sideslip angle; and *ctrl* refers to control surface deflection. The variable CS denotes the number of control surfaces present on the aircraft. For the SwanK1, it is important to note that $CS = 0$. AR refers to aspect ratio of the wing and e is the Oswald coefficient.

[5] `https://classic.gazebosim.org/tutorials?tut=aerodynamics`.

For the post-stall model, the lift and drag coefficients are calculated as:

$$C_L = 2\sin^2(\alpha)\cos(\alpha) + C_{Lp}p + C_{Lq}q + C_{Lr}r + \sum_{x=1}^{CS} C_{L,ctrl,x}\delta_{ctrl,x} + C_{L\beta}\beta, \tag{4.27}$$

$$C_D = C_{D,FP}\,(0.5 - 0.5\cos(2\alpha)) + C_{D\beta}\beta + C_{Dp}p + C_{Dq}q + C_{Dr}r + \sum_{x=1}^{CS} C_{D,ctrl,x}\delta_{ctrl,x}, \tag{4.28}$$

with $C_{D,FP}$ representing the flat-plate coefficient of drag, which is computed by:

$$C_{D,FP} = \frac{2}{1 + e^{K_1 + K_2 AR}}, \tag{4.29}$$

where K_1 and K_2 are empirical coefficients.

To blend the pre-stall and post-stall functions, a sigmoid function σ is used:

$$\sigma = \frac{1 + e^{-M(\alpha-\alpha_{stall})} + e^{M(\alpha-\alpha_{stall})}}{(1 + e^{-M(\alpha-\alpha_{stall})})(1 + e^{M(\alpha-\alpha_{stall})})}, \tag{4.30}$$

where M is an empirical coefficient.

Then, the equations for the coefficients of lift and drag are:

$$\begin{aligned} C_L &= (1-\sigma)(C_{L0} + C_{L\alpha}\alpha) + 2\sigma\sin^2(\alpha)\cos(\alpha) \\ &\quad + C_{Lp}p + C_{Lq}q + C_{Lr}r + \sum_{x=1}^{CS} C_{L,ctrl,x}\delta_{ctrl,x} + C_{Lb}\beta, \end{aligned} \tag{4.31}$$

$$\begin{aligned} C_D &= (1-\sigma)\left(C_{D0} + \frac{C_L^2}{\pi ARe}\right) + \sigma(C_{D,FP}\,(0.5 - 0.5\cos(2\alpha)) \\ &\quad + C_{Dp}p + C_{Dq}q + C_{Dr}r + C_{D\beta}\beta + \sum_{x=1}^{CS} C_{D,ctrl,x}\delta_{ctrl,x}). \end{aligned} \tag{4.32}$$

With the computation of the coefficients, Eqs. 4.15-4.20 are used to compute the Lift, Drag, Sideforce, Rolling, Pitching, and Yawing moments. This equations are similar to the ones seen in [36].

The aerodynamic coefficients used in the simulation for the quadrotor tailsitter are presented in Table 4.2.

To illustrate the computed lift coefficient by the *advanced_lift_and_drag_plugin* for

Parameter	**Value**	**Description**
S	0.15 m^2	Wing reference area
$\bar{c}$	0.987 m	Characteristic length
mac	0.22 m	Mean aerodynamic chord
AR	6.5	Aspect Ratio
e	0.97	Oswald Coefficient
C_{m0}	0.075	Pitching Moment Coefficient at zero AoA
C_{L0}	0.15188	Lift Coefficient at zero AoA
C_{D0}	0.029	Drag coefficient at zero angle of attack
C_{La}	5.015 rad^{-1}	Slope of C_L-alpha curve
C_{ma}	-0.463966 rad^{-1}	Pitching moment slope wrt alpha - before stall
C_{Yb}	-0.258244 rad^{-1}	Side force slope wrt beta
C_{lb}	-0.039250 rad^{-1}	Roll moment slope wrt beta
C_{nb}	0.100826 rad^{-1}	Yaw moment slope wrt beta
C_{Yp}	0.065861 rad^{-1}	Side force slope wrt roll rate
C_{Lq}	7.971792 rad^{-1}	Lift coefficient slope wrt pitching rate
C_{mq}	-12.140140 rad^{-1}	Pitching moment slope wrt pitching rate
C_{Yr}	0.230299 rad^{-1}	Side force slope wrt yaw rate
C_{lr}	0.078165 rad^{-1}	Roll moment slope wrt yaw rate
α_{stall}	0.3391 rad	Stall angle
$C_{La_{stall}}$	-3.85 rad^{-1}	Lift coefficient slope wrt angle of attack at stall
$C_{Da_{stall}}$	-0.9234	Drag coefficient at stall

Table 4.2: Aerodynamic parameters used in simulation.

a comparison with *lift_and_drag_plugin*, the angular velocities $\boldsymbol{\omega}$ and the sideslip angle β were set to zero and only the variation of the angle of attack α was considered. The computed lift coefficient C_L is shown in Fig. 4.3.

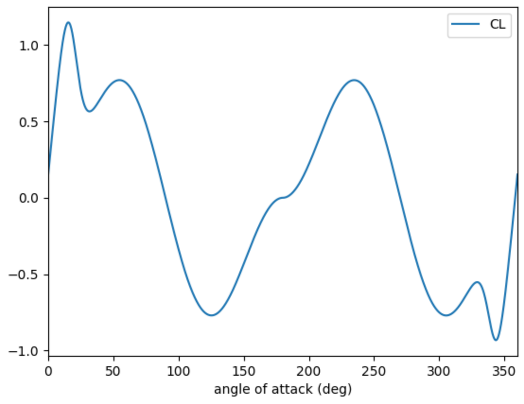


Figure 4.3: Computation of lift coefficient by *advanced_lift_drag_plugin*, only considering the effect of the AoA. The values are from the modeled quadrotor tailsitter.

Based on Fig. 4.3, it is seen that the *advanced_lift_and_drag* plugin incorporates nonlinearities instead of approximating the computation of the lift coefficient as two lines, as done in the *lift_and_drag_plugin*. Therefore, since in this dissertation the focus is on a tailsitter, a vehicle that flights at high AoAs, it was considered more appropriate to use the nonlinear model.

Chapter 5

Controllers Design

This chapter describes the control architecture, the recovery procedure, and the design of the two different controllers.

5.1 Architecture

Figure 5.1 summarizes the control architecture used. On the companion computer, the desired thrust T_d and torques $\boldsymbol{\tau}_d = \begin{bmatrix}\tau_{d,x} & \tau_{d,y} & \tau_{d,z}\end{bmatrix}^T$ can be sent either by the PID or NMPC controller, according to which controller is running. The flight controller receives the normalized values T_{norm} and $\boldsymbol{\tau}_{norm}$, which are computed by:

$$T_{norm} = \frac{\sqrt{\frac{4T_d}{c_t}} - \omega_{min}}{\omega_{max} - \omega_{min}} \tag{5.1}$$

$$\boldsymbol{\tau}_{norm} = \begin{bmatrix}\frac{\tau_{d,x}}{\tau_{x_{max}}} & \frac{\tau_{d,y}}{\tau_{y_{max}}} & \frac{\tau_{d,z}}{\tau_{z_{max}}}\end{bmatrix}^T \tag{5.2}$$

where c_t is the motor constant as seen in Eq. 4.7, ω_{max} and ω_{min} are, respectively, the maximum and minimum rotational velocity of each rotor, and $\tau_{x_{max}}$, $\tau_{y_{max}}$, and $\tau_{z_{max}}$ are, respectively, the maximum torques in the x, y, and z directions with respect to the body frame that all the propellers combined can provide to the UAV. These values are shown in Tab. 5.2. Notice that the controllers are designed such that the computed thrust is aligned with the body's $\hat{\mathbf{i}}_{\mathcal{B}}$ axis, although one could also include forces generated by the propellers in the other axes of the body frame since the propellers are slightly tilted.

The normalized thrust and torques are then passed to the flight controller by the top-

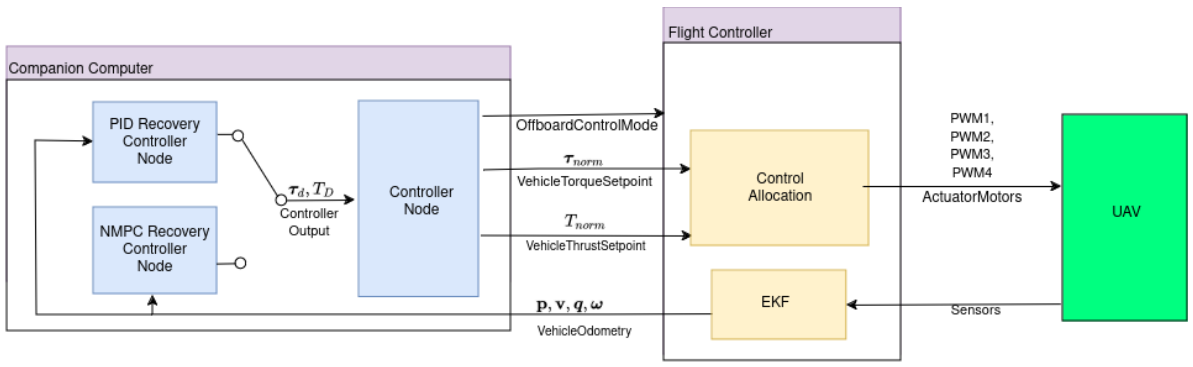


Figure 5.1: Control Architecture. The flight controller runs on offboard mode, receiving normalized commanded thrust T_{norm} and torques $\boldsymbol{\tau}_{norm}$ from the companion computer and providing the state estimate. The control allocation used is implemented in the flight controller and sends the individual commands to the rotors.

ics: *px4_msgs::msg::VehicleThrustSetpoint* [1] and *px4_msgs::msg::VehicleTorqueSetpoint* [2], respectively.

The offboard mode in PX4 allows the vehicle to obey position, acceleration, attitude, attitude rates or thrust/torque setpoints provided by some source external to the flight stack, such as a companion computer. To do so, in addition to the setpoints provided, the companion computer must also send a heartbeat signal of 2 Hz, which includes the type of setpoints provided via the OffboardMode message. In this dissertation, the thrust and torques are passed to the flight controller, bypassing the angular rate controller directly to the control allocation, also known as mixer. The control allocation, based on the desired thrust and torques provided, calculates the Pulse Width Modulation (PWM) signal that is send to control each one of the rotors individually. It takes the geometry properties of the UAV, particularly the position of each rotor and its rotation relative to the body frame, to define the output of each rotor.

The controllers receive from the Extended Kalman Filter (EKF) implemented in the PX4 the state estimation via the *vehicle_odometry* uORB message. It includes the position $\mathbf{p}$ in the inertial frame, the linear velocities $\mathbf{v}$ with respect to the inertial frame, the angular velocities $\boldsymbol{\omega}$ with respect to body frame, and the attitude $\boldsymbol{q}$ represented by an unitary quaternion. Since the implementation of the state estimator is out of the scope of this thesis, it is assumed that the output of the EKF provides a good state estimation and that the UAV is equipped with the necessary sensors, such as gyroscopes, inertial

[1]`https://docs.px4.io/main/en/msg_docs/VehicleThrustSetpoint.html`. Accessed 08-08-2024
[2]`https://docs.px4.io/main/en/msg_docs/VehicleTorqueSetpoint.html`. Accessed: 08-08-2024

measurement unit, barometer, magnetometer and Global Positioning System (GPS). As mentioned, the Gazebo simulator simulates each sensor, corrupting data by adding noise and bias to the measurements.

5.2 Recovery Procedure

The two proposed controllers divide the recovery process into two distinct stages. Since the goal of the recovery controller is to achieve a hovering condition, and given that the control principle of the quadrotor tailsitter is similar to that of a quadrotor—where heading control is limited, especially in hover—it is reasonable to first apply a reduced attitude control strategy. This approach focuses on controlling only the critical thrust direction in the initial phase. Then, after the UAV is close to the nominal attitude, its heading can be controlled. This increases the robustness of the system, as the axes corresponding to the inclination of the UAV are the ones responsible for the stability of the system [7].

Therefore, the first stage of the recovery focuses on adjusting the vehicle's attitude to the nominal orientation, disregarding its heading. Once the vehicle is sufficiently close to the pointing up attitude, within 10 degrees of alignment and angular velocities in the y and z body axes smaller than 8 rad/s, the second stage begins. In this step, the vehicle stabilizes its altitude while keeping the attitude pointing up, optionally aligning with a specified heading. If, for any reason, such as wind disturbances, the controller is unable to keep the attitude stable in the second stage, it goes back to the first one.

To achieve the desired behavior, it is necessary to split the attitude error into two different components: one related to the heading of the vehicle and the other to the inclination. Similar approaches are shown in [7, 30] for attitude control of tailsitter, however using rotations matrices to represent the attitude. In this dissertation, unit quaternions are used, due to the less number of parameters required, which leads to a controller mostly similar to [22]. The main difference are the reference frames used, since in [22], a quadrotor was used.

The hovering condition of the quadrotor tailsitter is such that the $\hat{\mathbf{i}}_{\mathcal{B}}$ is aligned and in the opposite direction of $\hat{\mathbf{k}}_{\mathcal{I}}$. Therefore, in the inertial frame $\mathcal{I}$, the desired thrust direction $\hat{\mathbf{e}}_{x,ref}$ is:

$$\hat{\mathbf{e}}_{x,ref} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}. \tag{5.3}$$

The UAV's current thrust direction $\hat{\mathbf{e}}_x$ is given by:

$$\hat{\mathbf{e}}_x = \begin{bmatrix} 2(q_0^2 + q_1^2) - 1 \\ 2(q_1 q_2 + q_0 q_3) \\ 2(q_1 q_3 - q_0 q_2) \end{bmatrix} \tag{5.4}$$

which is the same as getting the first column of the rotation matrix $\mathbf{R}$, representing the UAV's x-axis in the inertial frame (Eq. 2.32).

An error quaternion $\boldsymbol{q}_e$ that represents the necessary rotation to align these two vectors is then defined. The angle ϑ between the two vectors and a normal vector $\hat{\mathbf{n}}$ to both of them are calculated as:

$$\vartheta = \arccos(\hat{\mathbf{e}}_x \cdot \hat{\mathbf{e}}_{x,ref}) \tag{5.5}$$

$$\hat{\mathbf{n}} = \frac{\hat{\mathbf{e}}_x \times \hat{\mathbf{e}}_{x,ref}}{||\hat{\mathbf{e}}_x \times \hat{\mathbf{e}}_{x,ref}||}. \tag{5.6}$$

Since the objective is to apply this rotation with respect to the body frame, the rotation axis $\hat{\mathbf{n}}$ is represented in the body frame $\mathcal{B}$ coordinates by using the current estimate of attitude:

$$\hat{\mathbf{n}}^{\mathcal{B}} = \boldsymbol{q}^{-1} \otimes \hat{\mathbf{n}}, \tag{5.7}$$

where $\otimes$ represent the rotation performed by a quaternion (Eq. 2.32).

The error quaternion $\boldsymbol{q}_{e,inc}$ that computes the inclination error is then constructed as:

$$\boldsymbol{q}_{e,inc} = \begin{bmatrix} \cos(\frac{\vartheta}{2}) \\ \hat{\mathbf{n}}^{\mathcal{B}} \sin(\frac{\vartheta}{2}) \end{bmatrix}. \tag{5.8}$$

If $\vartheta = 0$, the rotation axis is undetermined and the error quaternion $\boldsymbol{q}_{e,inc}$ is set to the identity $\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T$. By construction, the x component of $\boldsymbol{q}_{e,inc}$, that is q_{e,inc_1}, is always zero, which assures that no rotation around the body's x axis is necessary to align $\hat{\mathbf{e}}_x$ with $\hat{\mathbf{e}}_{x,ref}$.

Considering the full attitude control, given the desired heading $\boldsymbol{q}_{head,des}$, the reference quaternion $\boldsymbol{q}_{ref}$ is computed as:

$$\boldsymbol{q}_{ref} = \boldsymbol{q}_{head,des} \circ \boldsymbol{q}_{hover} \tag{5.9}$$

where $\boldsymbol{q}_{hover}$ corresponds to the quaternion related to the hovering condition, that is, $\boldsymbol{q}_{hover} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \end{bmatrix}^T$, with a specified heading of 0 rad.

The quaternion $\boldsymbol{q}_e$ that represents the rotation necessary to drive $\boldsymbol{q}$ to $\boldsymbol{q}_{ref}$ is given by:

$$\boldsymbol{q}_e = \boldsymbol{q}_{ref} \circ \boldsymbol{q}^{-1} = \boldsymbol{q}_{ref} \circ \overline{\boldsymbol{q}}. \tag{5.10}$$

The error quaternion is divided into a heading $\boldsymbol{q}_{e,head}$ and a inclination part $\boldsymbol{q}_{e,inc}$:

$$\boldsymbol{q}_e = \boldsymbol{q}_{e,head} \circ \boldsymbol{q}_{e,inc}. \tag{5.11}$$

Thus, by using Eqs. 5.10 and 5.11, the final rotation $q_{e,head}$ around the body's x axis can then be computed as:

$$\boldsymbol{q}_{e,head} = \boldsymbol{q}_{ref} \circ (\boldsymbol{q}_{e,inc} \circ \boldsymbol{q})^{-1}. \tag{5.12}$$

By construction, the y and z components of $\boldsymbol{q}_{e,head}$, that is $q_{e,head_2}$ and $q_{e,head_3}$, are always zero.

5.3 Controllers design

This section describes the structure of the two controllers proposed: the PID and the NMPC.

5.3.1 Proportional, Integrative and Derivative Controller

The controller consists of two distinct PID controllers: one attitude controller, based on [22], and a separated altitude controller, based on [5]. The former is capable of operating in all flight modes indistinguishably, providing the reference torque to the vehicle, while the latter computes the desired thrust based on the altitude.

Attitude Controller

A unified attitude controller based on unitary quaternions is proposed. It consists of a cascaded control structure, as seen in Fig. 5.2.

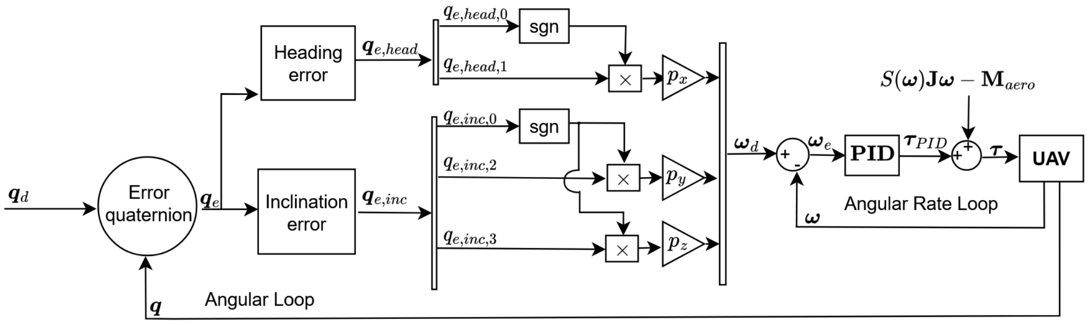


Figure 5.2: Attitude Controller structure.

The outer loop of the attitude controller is an angular loop and the inner loop is an angular rate loop. The former is a proportional controller, which uses unit quaternions to represent the attitude and calculates the desired angular rates $\boldsymbol{\omega}_d$.

From Eqs. 5.8 - 5.12, the quaternions corresponding to the inclination $\boldsymbol{q}_{e,inc}$ and the heading $\boldsymbol{q}_{e,head}$ are calculated. From the former, the desired angular rates $\omega_{d,y}$ and $\omega_{d,z}$, around the bodies y and z axes respectively, can be computed as:

$$\omega_{d,y} = \begin{cases} 2\,p_y\,q_{e,inc_2} & \text{if } q_{e,inc_0} \geq 0 \\ -2\,p_y\,q_{e,inc_2} & \text{if } q_{e,inc_0} < 0 \end{cases} \tag{5.13}$$

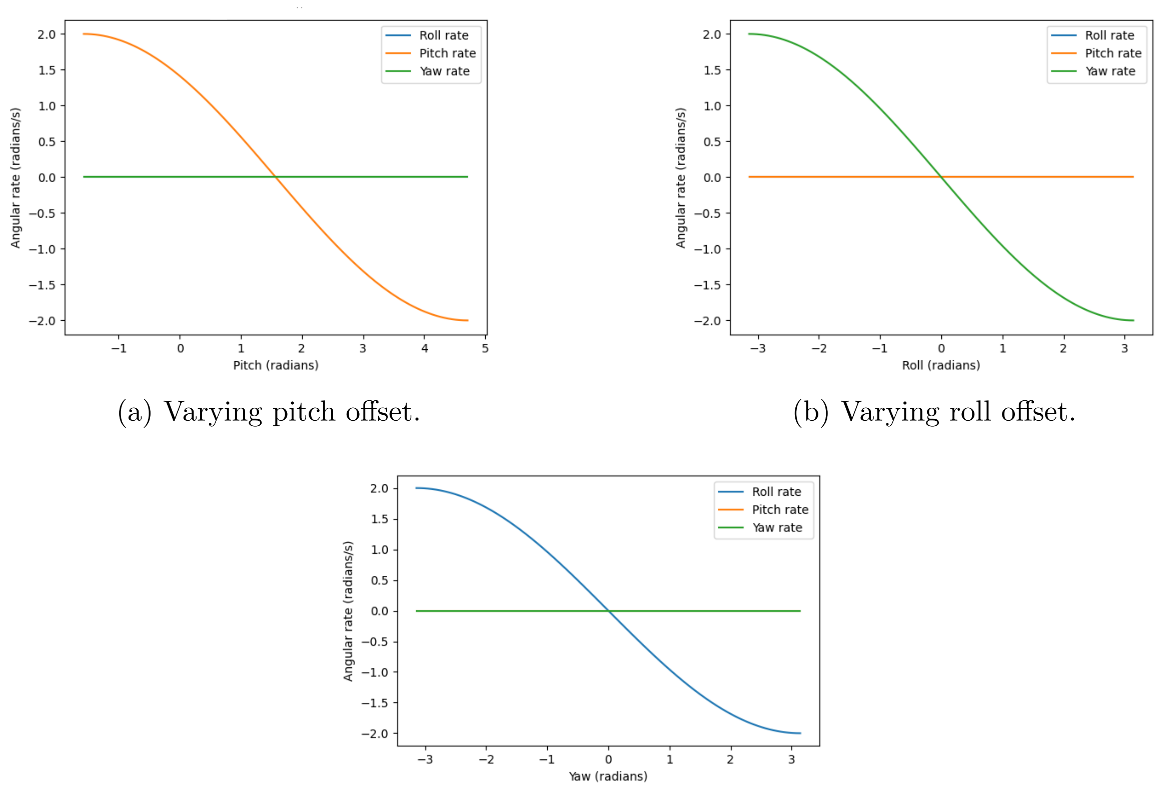
$$\omega_{d,z} = \begin{cases} 2\,p_z\,q_{e,inc_3} & \text{if } q_{e,inc_0} \geq 0 \\ -2\,p_z\,q_{e,inc_3} & \text{if } q_{e,inc_0} < 0 \end{cases} \tag{5.14}$$

where p_y and p_z represent tuning parameters corresponding to a proportional controller. This control law is shown to be globally asymptotically stable and its discrete implementation is robust to measurement noise [42].

Similarly, the desired body rate $\omega_{d,x}$ around the body's x axes can be computed from $\boldsymbol{q}_{e,head}$ as:

$$\omega_{d,x} = \begin{cases} 2\,p_x\,q_{e,head_1} & \text{if } q_{e,head_0} \geq 0 \\ -2\,p_x\,q_{e,head_1} & \text{if } q_{e,head_0} < 0 \end{cases}. \tag{5.15}$$

Splitting the attitude error into the inclination $q_{e,inc}$ and heading $q_{e,head}$ allows to control first the inclination of the UAV and only after it relatively stable, control the



(a) Varying pitch offset.

(b) Varying roll offset.

(c) Varying yaw offset.

Figure 5.3: Error function properties. Desired angular rate $\boldsymbol{\omega}_d$ for varying one of the ZXY Euler angles, while keeping the others constant. The reference is the quaternion corresponding to hover.

heading, which is desirable for the recovery control procedure and due to different control limits when hovering. To validate that the computed error captures the desired heading error and inclination error, the plots in Fig. 5.3 were generated. They show how each component of the desired angular rate changes for a variation in one of the Euler ZYX angles while the others remain constant at the reference. The reference is the quaternion that corresponds to the hovering condition with zero yaw.

As seen by Fig. 5.3, the desired behaviour is obtained. When there is a change in one of the Euler angles, the desired angular rate is computed such that the it is will drive the Euler angle error to zero. Notice that since ZXY Euler angles are used, the yaw offset is compensated by ω_x (roll rate), while the roll offset is compensated by ω_z (yaw rate).

The inner loop of the attitude controller is a PID controller with feedforward terms to cancel the aerodynamic moments $\mathbf{M}_a$ and the Coriolis term $S(\boldsymbol{\omega})\mathbf{J}\boldsymbol{\omega}$. The output of the inner loop controller is the desired torque $\boldsymbol{\tau}_d$, given by:

$$\boldsymbol{\tau}_d = \mathbf{K_P}\boldsymbol{\omega}_e + \mathbf{K_I}\int \boldsymbol{\omega}_e \, dt + \mathbf{K_D}\frac{d\boldsymbol{\omega}_e}{dt} + S(\boldsymbol{\omega})\mathbf{J}\boldsymbol{\omega} - \mathbf{M}_a \tag{5.16}$$

where $\mathbf{K_P}, \mathbf{K_I}, \mathbf{K_D} \in \mathbb{R}^{3\times 3}$ are positive diagonal matrices, representing the gains for the PID controller. The angular velocity error $\boldsymbol{\omega}_e$ is given by:

$$\boldsymbol{\omega}_e = \boldsymbol{\omega}_d - \boldsymbol{\omega} \tag{5.17}$$

where $\boldsymbol{\omega}$ is the current angular velocity of the UAV and $\boldsymbol{\omega}_d = \begin{bmatrix}\omega_{d,x} & \omega_{d,y} & \omega_{d,z}\end{bmatrix}^T$.

The feedforward term corresponding to the aerodynamic moment is calculated based on Eqs. 4.24 - 4.26, but only taking into account the contributions of the AoA and the sideslip angle, thus neglecting the terms corresponding to the angular velocity.

Altitude Controller

The altitude controller is designed to hold the UAV altitude at a desired level. It takes as input a desired altitude h_d and based on the current altitude of the vehicle h, computes a reference thrust T_d in the direction of the body's x axis. Taking into account the mathematical model of the UAV, particularly, Eqs. 4.1 and 4.2:

$$\ddot{h} = g + \frac{1}{m}\mathbf{r_3}(\mathbf{F}_a + \mathbf{F}_r) \tag{5.18}$$

where $\mathbf{r_3}$ indicates the third row of $\mathbf{R}$, $\mathbf{F}_a$ and $\mathbf{F}_r$ are, respectively, the forces produced by the aerodynamic surfaces and rotors.

Based on Eq. 5.18, a PID controller is designed to compute the desired thrust as follows:

$$\mathbf{r_3}(\mathbf{F}_a + \mathbf{F}_r) + mg = k_p h_e + k_i \int h_e dt + k_d \frac{dh_e}{dt} \tag{5.19}$$

where k_p, k_i, and k_d are the gains of the PID controller and $h_e = h_d - h$ represents the altitude error.

The resulting thrust desired force T_d, which is the first component of $\mathbf{F}_r$, is therefore:

$$T_d = \frac{k_p h_e + k_i \int h_e dt + k_d \frac{dh_e}{dt} - mg - \mathbf{r_3}\mathbf{F}_a}{r_{31}} \tag{5.20}$$

where r_{31} is the first element of $\mathbf{r_3}$. The computed $\mathbf{F}_a$ only considers the effects of the AoA and sideslip angle, thus neglecting the influence of the angular velocities in the computation of the forces. Notice that this simple controller only attempts to stabilize the altitude of the vehicle.

To avoid the effects of windup in the integral terms of the controllers, the integral term is only computed and updated when the control effort remains within actuator limits. Once the control signal reaches saturation (maximum or minimum allowed value), the integration process halts, and the previous integral value is maintained. Additionally, low pass filters are implemented to the signals passed to the derivative terms to avoid amplifying noise.

For the recovery procedure, there is no prior knowledge of the altitude of the vehicle. Therefore, when the UAV is still stabilizing itself to a hovering condition, the PID term of the altitude controller is turned off. This makes the quadrotor tailsitter try to only compensate for the gravity and the computed aerodynamic forces. Once the vehicle is close to the nominal inclination, the current altitude is set as the desired height h_d and the PID controller is turned on.

Parameters

The parameters of the PID controller were obtained through manual tuning. The values obtained are shown in Table 5.1

Parameter	Value	Description	Controller
p_x	0.3	Proportional gain, x-axis	Attitude
p_y	0.65	Proportional gain, y-axis	Attitude
p_z	2.0	Proportional gain, z-axis	Attitude
$\mathbf{K_P}$	diag(0.1, 0.15, 0.5)	Proportional Gain	Angular Rate
$\mathbf{K_I}$	diag(0.1, 0.2, 0.2)	Integral Gain	Angular Rate
$\mathbf{K_D}$	diag(0.1, 0.1, 0.1)	Derivative Gain	Angular Rate
k_p	0.6	Proportional Gain	Altitude
k_i	0.9	Integral Gain	Altitude
k_d	0.2	Derivative Gain	Altitude

Table 5.1: PID controller parameters.

5.3.2 Nonlinear Model Predictive Controller

To design the NMPC, it is necessary to define the states and the inputs, the constraints, the cost function as well as the prediction model.

States and Inputs

Considering the dynamic model given by Eqs. 4.1-4.4, the following state vector $\mathbf{x}$ and control input vector $\mathbf{u}$ are used:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} & \mathbf{v} & \mathbf{q} & \boldsymbol{\omega} \end{bmatrix}^T \tag{5.21}$$

$$\mathbf{u} = \begin{bmatrix} T & \tau_x & \tau_y & \tau_z \end{bmatrix}^T . \tag{5.22}$$

Here, the state vector $\mathbf{x}$ includes the position $\mathbf{p}$, velocity $\mathbf{v}$, quaternion attitude $\boldsymbol{q}$, and angular velocity $\boldsymbol{\omega}$. The control input vector $\mathbf{u}$ consists of the thrust T and the torques $\boldsymbol{\tau}$ applied in the body frame.

Objective Function

One of the key features to design the NMPC is to determine the objective function J to be minimized:

$$J = ||\mathbf{e}_{x_N}||^2_{\mathbf{Q}} + \sum_{i=0}^{N-1} \left(||\mathbf{e}_{x_i}||^2_{\mathbf{Q}} + ||\mathbf{e}_{u_i}||^2_{\mathbf{P}} \right) \tag{5.23}$$

which includes penalization at each predicted instant j on the inputs error $\mathbf{e}_{u_j}$, $j = 0, \cdots N$, and on the reference state error $\mathbf{e}_{x_i}$. The penalizations are weighted by diagonal matrices $\mathbf{P} = \text{diag} \begin{bmatrix} p_T & \mathbf{p}_\tau \end{bmatrix}$, which corresponds to the weights on the thrust T and torques $\boldsymbol{\tau}$ errors, and $\mathbf{Q} = \text{diag} \begin{bmatrix} \mathbf{q}_p & \mathbf{q}_v & \mathbf{q}_q & \mathbf{q}_\omega \end{bmatrix}$ corresponding to weights on the state error.

The control input error $\mathbf{e}_{u_i}$ is calculated as:

$$\mathbf{e}_{u_i} = \begin{bmatrix} T_i - T_i^{\text{ref}} \\ \boldsymbol{\tau}_i - \boldsymbol{\tau}_i^{\text{ref}} \end{bmatrix} \tag{5.24}$$

whereas the state error $\mathbf{e}_{x_j}$ is defined by:

$$\mathbf{e}_{x_j} = \begin{bmatrix} \mathbf{p}_j - \mathbf{p}_j^{\text{ref}} \\ \mathbf{v}_j - \mathbf{v}_j^{\text{ref}} \\ q_{j,yz} \\ q_{j,x} \\ \boldsymbol{\omega}_j - \boldsymbol{\omega}_j^{\text{ref}} \end{bmatrix}, j = 0, \cdots, N \tag{5.25}$$

where $q_{j,xy}$ and $q_{j,z}$ represent penalties on the quaternion error at each instant j. The

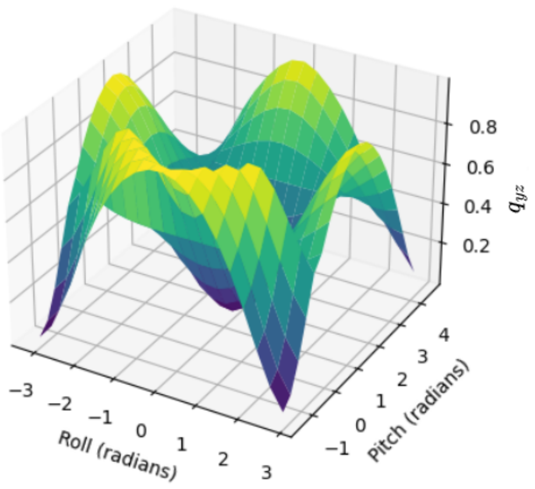


Figure 5.4: Error q_{yz} for constant yaw of zero rad and varying pitch and roll using Euler ZXY sequence and the quaternion corresponding to hover as reference.

control input reference corresponds to the hovering condition: $T_i^{\text{ref}} = mg$ and $\boldsymbol{\tau}_i^{\text{ref}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$. $\mathbf{p}_j^{\text{ref}}$ is only defined when the vehicle reaches the second stage of the recovery procedure, $\mathbf{v}_j^{\text{ref}}$ and $\boldsymbol{\omega}_j^{\text{ref}}$ are always zero. The reference attitude is given by Eq. 5.9. The attitude error is split into two parts, one corresponding to the heading $\boldsymbol{q}_{e,head}$ and another to the inclination $\boldsymbol{q}_{e,inc}$, as seen by Eqs. 5.5-5.12. This concept is also used in NMPC formulation of [26], in which this extraction is done to consider the lack of yaw control in a quadrotor if one of the rotors breaks.

With the two quarternions $\boldsymbol{q}_{head}$ and $\boldsymbol{q}_{inc}$ calculated at each instant j, the cost of the attitude error is given by:

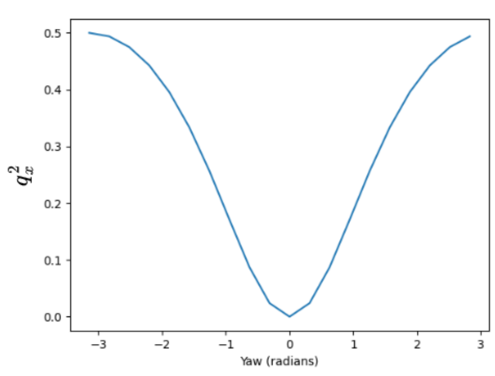
$$q_{j,yz} = q_{e,inc_2}^2 + q_{e,inc,3}^2 \tag{5.26}$$

$$q_{j,x} = q_{e,head_1} \tag{5.27}$$

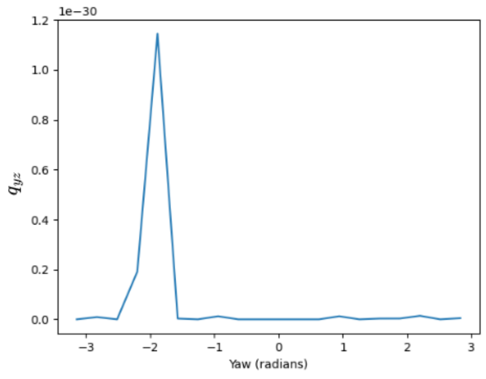
where q_{e,inc_1} and q_{e,inc_2} corresponds to the elements in index one and two of $\boldsymbol{q}_{e,inc}$ and $q_{e,head3}$ is the last element of $\boldsymbol{q}_{e,head}$.

With these calculations, it is possible to penalize differently the error in the heading of the UAV and its alignment with the horizontal plane. Figure 5.4 shows the error q_{yz} for a constant yaw of 0 rad and varying pitch and roll using Euler ZXY sequence. The quaternion reference is set as the hovering condition: $\boldsymbol{q}_{\text{hover}} = \begin{bmatrix} \sqrt{2}/2 & 0 & \sqrt{2}/2 & 0 \end{bmatrix}^T$, which corresponds to 0 rad in roll, $\frac{\pi}{2}$ in pitch and 0 rad in yaw.

Furthermore, the errors q_x and q_{yz} are analyzed for a varying yaw and constant pitch and roll, which are set to $\frac{\pi}{2}$ rad and zero rad, respectively. The reference is set as the hovering quaternion $\boldsymbol{q}_{hover}$ and the results are shown in Fig. 5.5.



(a) Squared heading error q_x^2.



(b) Inclination error q_{yz}.

Figure 5.5: Squared heading error q_x^2 and inclination error q_{yz} for constant pitch and roll and varying yaw. The reference is set as the quaternion that corresponds to hovering with yaw equal to zero.

As seen from Fig. 5.4, the inclination error does not change for a varying yaw and from Fig. 5.5, it tends to zero as the pitch and roll tends to the reference. Additionally, from Fig. 5.5a, it is seen that the heading error captures the difference in the yaw, while maintaining the inclination error equal to zero for a pitch of $\frac{\pi}{2}$ rad and a roll of 0 rad.

Prediction Model

The prediction model of the NMPC is based on Eqs. 4.24 - 4.26. To reduce the computation load, following the approach seen in [35], the part of the drag coefficient C_D that is dependent on the angle of attack α was fitted into a polynomial of order 6, $C_D(\alpha)$. The polynomial expression is:

$$\begin{aligned} C_D(\alpha) =& 1.5265 \times 10^{-2}\alpha^6 + 4.7152 \times 10^{-05}\alpha^5 - 2.6015 \times 10^{-01}\alpha^4 - 5.2242 \times 10^{-04}\alpha^3 \\ & + 1.0973\alpha^2 + 1.2426 \times 10^{-03}\alpha^1 + 8.514810^{-02} \end{aligned} \tag{5.28}$$

Figure 5.6 shows the comparison between the fitted curve and the the C_D as calculated by the *advanced_lift_drag_plugin*, considering only the effect of the AoA, making the other values, such as the sideslip angle β and the angular velocity $\boldsymbol{\omega}$ zero. The mean square error is 0.5232. This is done to provide a simpler expression for the computed drag.

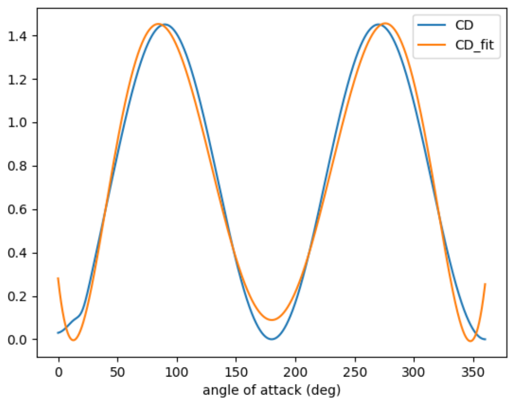


Figure 5.6: Comparison between C_D calculated by *advanced_lift_drag_plugin* by considering only the effect of α and the fitted polynomial curve of order 6.

To make the expressions of the aerodynamic coefficients similar to what is obtained with data from wind tunnel measurements, the expressions for the aerodynamic coefficients neglect the effects of the angular velocity $\boldsymbol{\omega}$, such that:

$$C_D = C_D(\alpha) + C_{D\beta}\beta \tag{5.29}$$

$$C_L = (1-\sigma)(C_{L0} + C_{L\alpha}\alpha) + 2\sigma \sin^2(\alpha)\cos(\alpha) + C_{L\beta}\beta \tag{5.30}$$

$$C_Y = C_{Y\alpha}\alpha + C_{Y\beta}\beta \tag{5.31}$$

where it was already considered that the number of control surfaces is zero for the quadrotor tailsitter.

Similarly, the expressions for the coefficients corresponding to the aerodynamic moments are simplified as:

$$C_l = C_{l\alpha}\alpha + C_{l\beta}\beta \tag{5.32}$$

$$C_m = C_{m0} + C_{m\alpha}\alpha + C_{m\beta}\beta \tag{5.33}$$

$$C_n = C_{n\alpha}\alpha + C_{n\beta}\beta. \tag{5.34}$$

Constraints

Constraints are given relative to the inputs:

$$0 \leq T \leq T_{max} \tag{5.35}$$

$$-\tau_{x_{max}} \leq \tau_x \leq \tau_{x_{max}} \tag{5.36}$$

$$-\tau_{y_{max}} \leq \tau_y \leq \tau_{y_{max}} \tag{5.37}$$

$$-\tau_{z_{max}} \leq \tau_z \leq \tau_{z_{max}} \tag{5.38}$$

which are calculated based on the geometry and properties of the quadrotor tailsitter. Notice that the minimum thrust is set to zero as the rotors only rotate in one direction. The constraint values are shown in Table 5.2.

Input	Value	Description
T_{max}	26.05 N	Maximum Thrust
$\tau_{x_{max}}$	1.548 Nm	Maximum torque in $\hat{\mathbf{i}}_{\mathcal{B}}$
$\tau_{y_{max}}$	3.468 Nm	Maximum torque in $\hat{\mathbf{j}}_{\mathcal{B}}$
$\tau_{z_{max}}$	5.501 Nm	Maximum torque in $\hat{\mathbf{k}}_{\mathcal{B}}$

Table 5.2: Input constraints.

Implementation

The dynamic model was discretized into N steps over the time horizon T of size dt = T/N using the Runge–Kutta fourth order method. The NMPC was implemented using CasADi, an open-source tool for nonlinear optimization and algorithmic differentiation [43]. The parameters of the controller used in the simulations are shown in Table 5.2.

Parameter	Value
$\mathbf{q}_p$	$\begin{bmatrix}8.0 & 8.0 & 150.0\end{bmatrix}$
$\mathbf{q}_v$	$\begin{bmatrix}3.0 & 3.0 & 40.0\end{bmatrix}$
$\mathbf{q}_q$	$\begin{bmatrix}4.0 & 800.0\end{bmatrix}$
$\mathbf{q}_\omega$	$\begin{bmatrix}7.0 & 7.0 & 7.0\end{bmatrix}$
p_T	0.5
$\mathbf{p}_\tau$	$\begin{bmatrix}3.0 & 3.0 & 3.0\end{bmatrix}$

Table 5.3: Parameters of NMPC controller.

Chapter 6

Simulation Results

This chapter presents and compares the results obtained from the two different approaches. Firstly, the metrics used for comparison are outlined, followed by the presentation and discussion of the simulation results.

6.1 Metrics

The objective is to evaluate and compare the performance of the recovery controllers implemented. The performance indications can be divided into the height drop to recovery, the time it takes to stabilize the UAV, the final velocities, and the robustness to wind gust.

The simulations are conducted such that the quadrotor tailsitter is launched in the air with different sets of initial velocity and attitude. This is done by applying a force and a torque to the vehicle at the beginning of the simulation, during which the controllers are turned off. Once the forces and torques are applied, the controllers are initiated, starting the recovery procedure. In the plots shown in this section, the time zero corresponds to the start of the recovery procedure. For an intuitive sense of attitude, instead of quaternions, the attitude is represented by the ZXY Euler angles. The reference for pitch and roll are always defined as $\frac{\pi}{2}$ and 0 rad, respectively, which corresponds to the hovering condition a heading of 0 rad. The reference yaw is not set directly, only the desired angular velocity corresponding to the heading is set to zero when the vehicle is close to the horizontal attitude. For a better visualization of the commanded thrust T and torques $\boldsymbol{\tau}$, the values are normalized by the maximum values that the quadrotor tailsitter can achieve. Notice that these normalized values can be greater than 1 or smaller than -1 if the controller does not take into account the actuators limits. The background color identifies the recovery steps: the first step is shown in white, while the second step is depicted in grey.

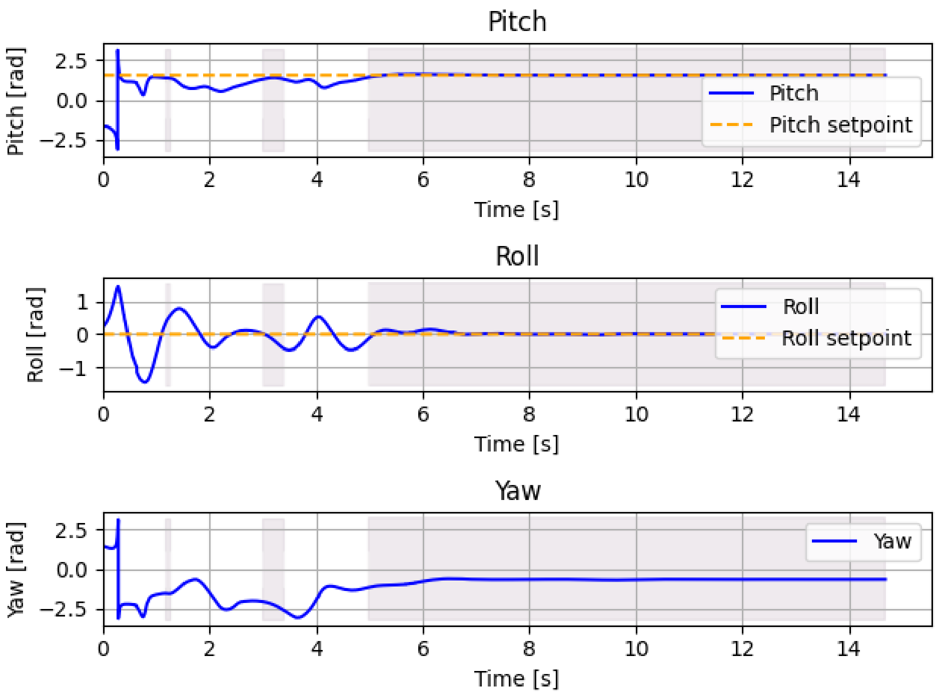
Both controllers simulated should provide thrust and torque setpoints at a frequency of 100 Hz. However, the designed NMPC remains too slow for real-time implementation, leading to a reduction in simulation speed. Through various experiments, it has been determined that the average operating rate of the NMPC is approximately 20 Hz. This performance may be improved by utilizing a dedicated computer for optimization while employing another for simulation. Additionally, leveraging C-code generation capabilities could enhance the computational efficiency of the controller, which was originally designed in Python. According to CasADi's documentation, "As a rule of thumb, the numerical evaluation of auto generated code, compiled with code optimization flags, can be between 4 and 10 times faster than the same code executed in CasADi's virtual machines". [1].

6.2 Upset Recovery

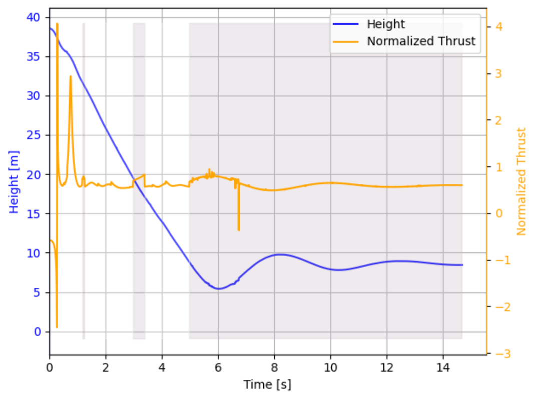
The UAV is thrown in the air such that the vehicle is upside-down when the controllers are activated. At the beginning of the recovery, the vehicle has a a velocity of 0.8 m/s directed to the ground and an initial altitude of 42 m. The results for the PID and NMPC controller are shown in Figs. 6.1 and 6.2, respectively. The videos of the upset recovery can be seen in `https://drive.google.com/drive/folders/1QZm7SAWj2Qb2LtJI5Tl9kYI8xYc-HYru?usp=sharing`.

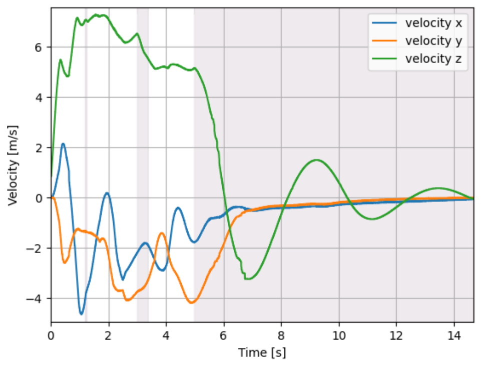
It is seen that both controllers are able to recover the quadrotor tailsitter from an upset initial condition. The final velocity of the vehicle tends to zero in both cases. It is seen that the vehicle achieves the second stage of the recovery much faster with the NMPC controller, in only 0.66 s compared to 5.0 s. The PID controller attempts to hover at a fixed position three different times, but is only capable of doing so in the last one. In the others, the attitude changes relatively fast and the vehicle needs to retry the recovery. Furthermore, the maximum height drop is of 6.41 m for the NMPC and 33.14 m for the PID. Notice that the final velocity in both cases are not exactly zero, however they were considered low enough to determine that the recovery was successful. Finally, it is seen that while the NMPC is able to respect the limits of thrust and torque, the PID fails to do so for the commanded thrust.

[1]`https://web.casadi.org/docs/`. Accessed: October 2024



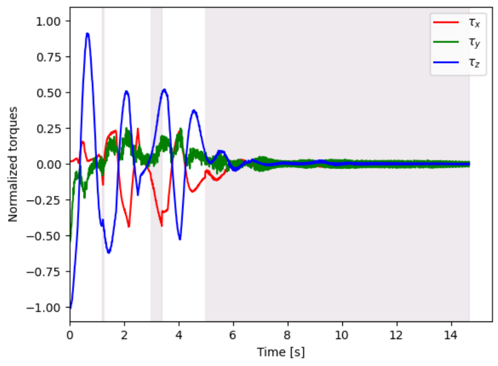
(a) Attitude Euler ZXY.





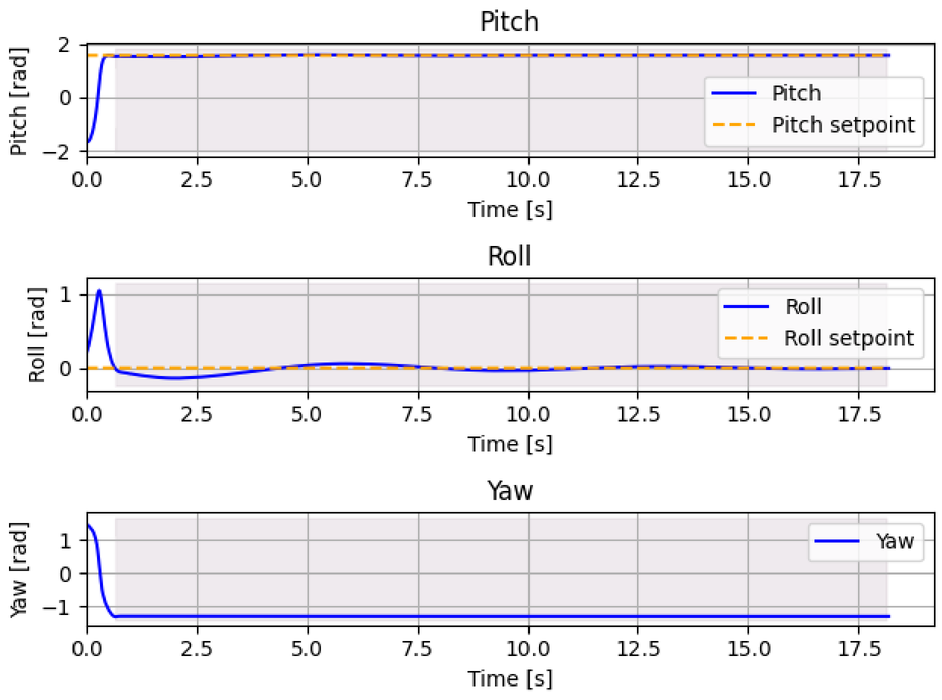
(b) Height and thrust over time.

(c) Velocity over time.

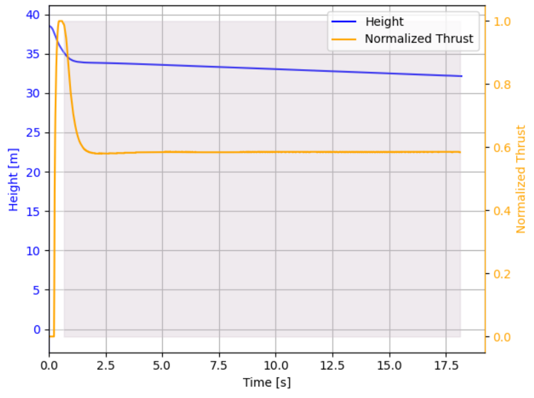


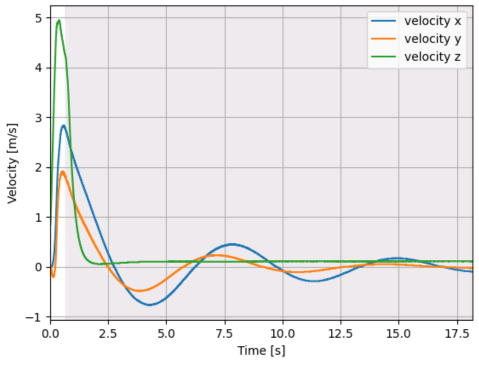
(d) Normalized Torques.

Figure 6.1: PID controller, upset recovery. In white, the controller stabilizes the attitude, in grey, the controller locks a position to hover. The control inputs are normalized relative to their respective limits, derived from the UAV parameters.



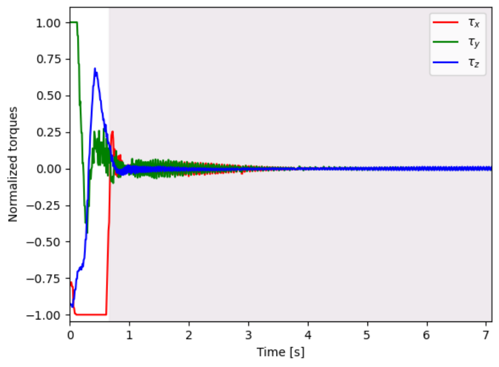
(a) Attitude Euler ZXY.





(b) Height and thrust over time.

(c) Velocity over time.



(d) Normalized Torques.

Figure 6.2: NMPC controller, upset recovery. In white, the controller stabilizes the attitude, in grey, the controller locks a position to hover. The control inputs are normalized relative to their respective limits, derived from the UAV parameters.

6.3 High speed recovery

Since the objective is to drop the vehicle from a mothership, it is important to assess the performance of the controllers when the initial velocity is significant. To test this scenario, the initial velocity is set to 22.5 m/s in the y direction, and zero in the others. The initial altitude is of 48.85 m and the initial attitude in ZXY Euler Angles is roll: 0.096 rad, pitch: -2.41 rad, and yaw: 1.86 rad, which corresponds to an inclination angle error ϑ_{init} of 2.26 rad.

The results for the PID controller and for the NMPC are shown in Figs. 6.3 and 6.4, respectively. The videos of the fast recovery are seen in `https://drive.google.com/drive/folders/1NecLDGXgAoDEhA3VAvYsrDBgXAIVVTI5?usp=sharing`.

From Figs. 6.3 and 6.4, it is seen that both controllers are able to stabilize attitude of the vehicle under a high initial velocity. For the PID controller, a hovering position is locked after 2.37 s and the maximum height drop is of 21.53 m. Notice that the velocity of the vehicle in the x and y directions tend to zero even though there is no direct control for the positions in these axes. The NMPC attempts to lock a hovering position after 1.4 s, however the hovering attitude is lost and another attempt is made at 4.6 s. The maximum height drop is of 12.45 m and the limits on the control inputs are taken into account.

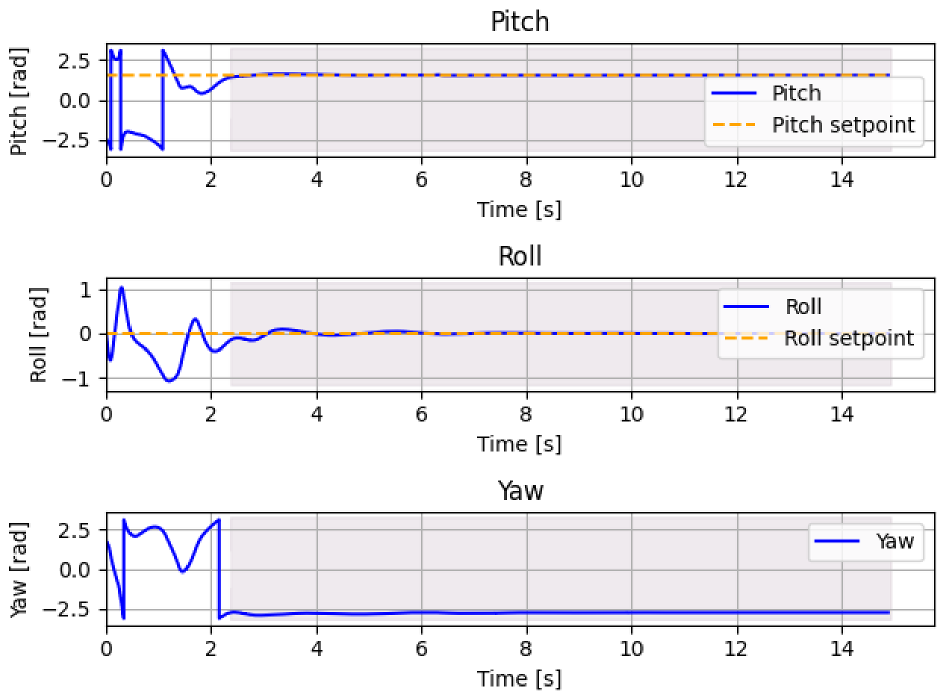
6.4 Windy Environment

To analyse the robustness of the controllers due to wind disturbances, winds were introduced. The controllers do not have an estimate of the wind velocity and, therefore, only the vehicle's velocity is used to compute the airspeed $\mathfrak{u}$, such that the wind speed relative to the Inertial frame $\mathbf{w}$ is zero in Eq. 4.9.

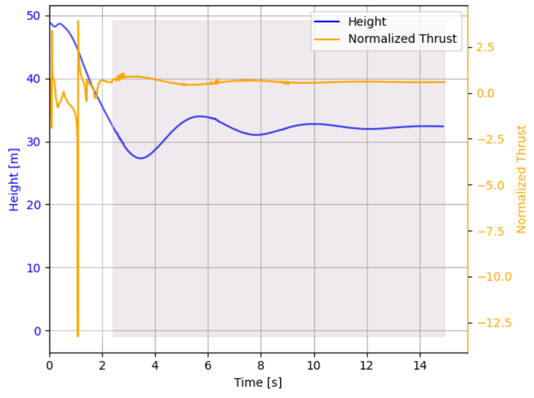
The wind is simulated using the wind plugin built in Gazebo [2]. It uses normal distributions to generate wind velocity and direction based on the mean μ and variance σ parameters. The wind strength and direction are randomly sampled every 0.5 s in simulated time, as an attempt to mimic the randomness of wind in real-world environments.

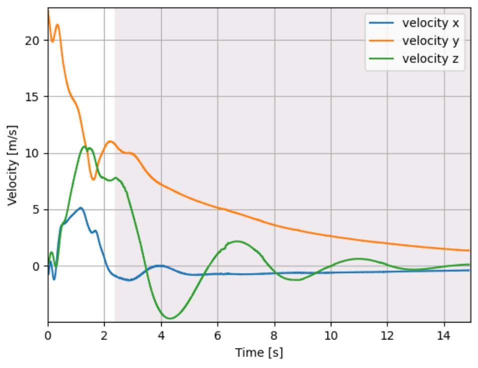
Various simulations were done with different sets of initial attitude, defined by the angular error ϑ_{init}, velocity $\mathbf{v}_{init}$ and different wind distributions, which are presented in Tab. 6.1. The sets 1-4 were generated by considering that the vehicle has an initial

[2]`https://github.com/PX4/PX4-SITL_gazebo-classic/blob/main/src/gazebo_wind_plugin.cpp`. Accessed: October 2024.



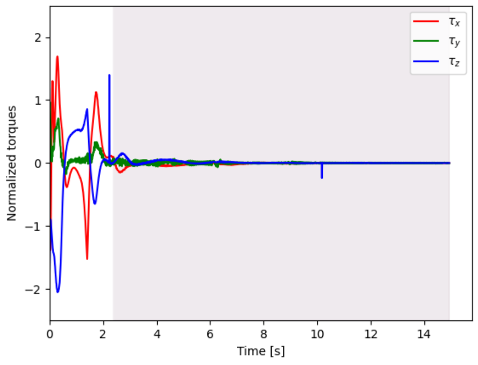
(a) Attitude Euler ZXY.





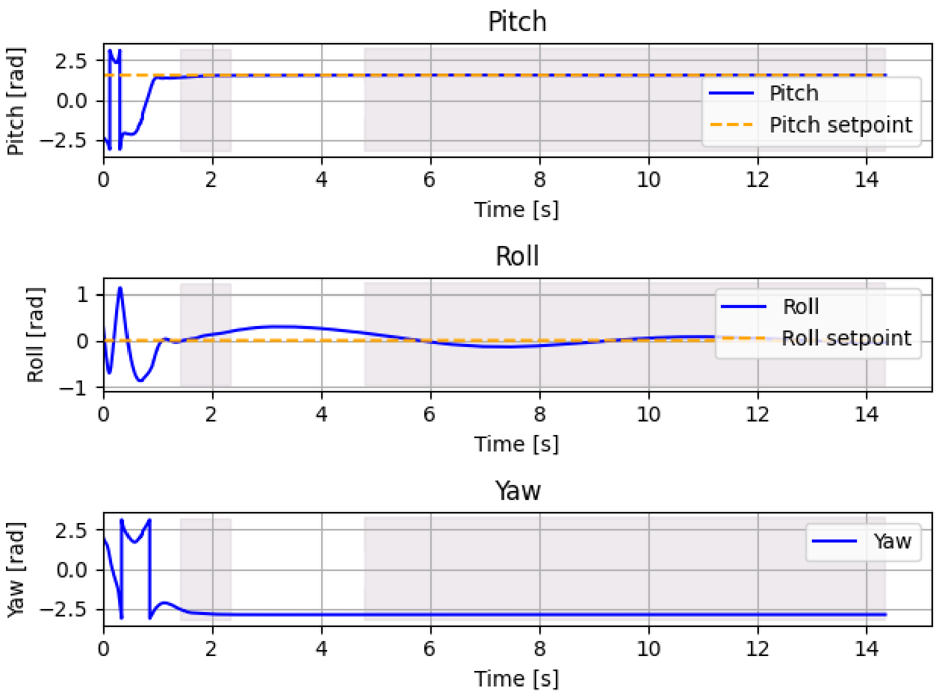
(b) Height and thrust over time.

(c) Velocity over time.

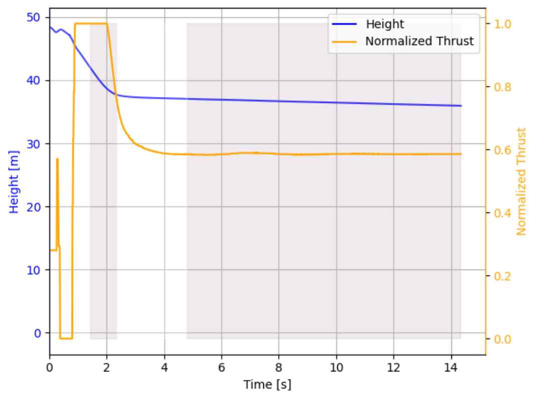


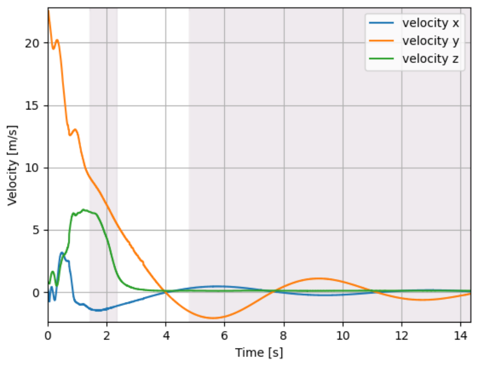
(d) Normalized Torques.

Figure 6.3: PID controller, with initial high velocity. In white, the controller stabilizes the attitude, in grey, the controller locks a position to hover. The control inputs are normalized relative to their respective limits, derived from the UAV parameters.



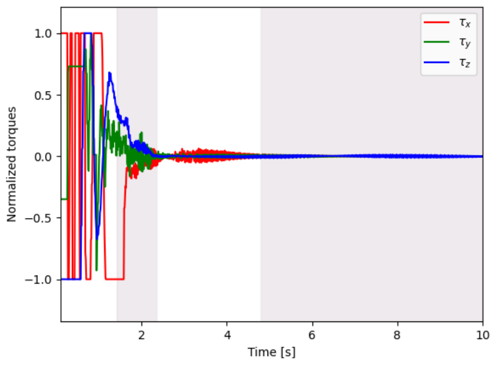
(a) Attitude Euler ZXY.





(b) Height and thrust over time.

(c) Velocity over time.



(d) Normalized Torques.

Figure 6.4: NMPC controller, with high initial velocity. In white, the controller stabilizes the attitude, in grey, the controller locks a position to hover. The control inputs are normalized relative to their respective limits, derived from the UAV parameters.

attitude ϑ_{init} such that the UAV is $\pi/2$ with respect to the hovering condition, that is, in a position similar to the horizontal flight. The sets 5-6 are generated by setting a considerable initial velocity in the y direction $v_{y,init}$ and assuming ϑ_{init} of 0.5 rad. In all simulations, the initial velocity of the vehicle in the z axis $v_{z,init}$ is positive, which means that the vehicle is moving towards the ground as it would happen in a drop from a mothership.

Table 6.1: Initial attitude, velocity and wind conditions for each simulation set.

Set	$\vartheta_{init}[rad]$	$v_{x,init}[m/s]$	$v_{y,init}[m/s]$	$v_{z,init}[m/s]$	$\mu_{wind}[m/s]$	σ_{wind} [m/s]
1	1.57	0.0	0.0	0.7	1.0	1.0
2	1.57	0.0	0.0	0.7	3.0	1.0
3	1.57	0.1	0.0	0.8	6.0	1.0
4	1.57	0.0	0.2	0.9	10.0	1.0
5	0.5	0.0	18.0	0.8	3.0	1.0
6	0.5	0.0	18.0	0.8	5.0	1.0
7	0.5	0.0	18.0	0.8	7.0	1.0

Each set of initial conditions and wind distributions was simulated ten times for each controller. The results obtained are summarized in Tab. 6.2. For a visual representation, some of the data is summarized in Figs. 6.5 and 6.6 for sets 1-4 and 5-7, respectively. For graphs displaying the average values, the corresponding standard deviation is also shown. The data is also shown with the standard deviation in the figures mentioned. The attitude recovery is considered successful if the vehicle maintains the inclination angle error ϑ smaller than 10° for a continuous period of at least 3 s and if it does not hit the ground. If this condition is not met, the controller is deemed to have failed. Notice that in this approach the success is not directly related to the stabilization in the height of the vehicle, although both controllers indeed try to maintain an altitude fixed during the second step of the recovery. For successful recoveries, the time to hold the hovering position t_{hold} is defined as the moment when the inclination angle error first remains below 10° for the continuous 3 s period. The average of t_{hold} for successful recoveries is depicted in the column $\bar{t}_{hold}$. The average height drop $\overline{h}_{drop}$ is also only calculated for successful recoveries and considers the initial altitude (set to 60 m) and the minimum altitude achieved during the recovery. The mean of the absolute velocities in all three axes during the period when the position is locked is also reported in the columns $\overline{|v_{x,\text{hold}}|}$, $\overline{|v_{y,\text{hold}}|}$, and $\overline{|v_{z,\text{hold}}|}$. All the simulations are limited to 15 seconds in simulated time.

By analysing the results for the sets 1-4, it is clear that the percentage of successful recoveries in both controllers decreases as the wind intensity becomes larger and, therefore,

Table 6.2: Results for the PID and NMPC controllers for different sets of initial attitude, velocity and wind conditions.

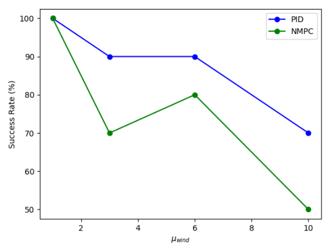
Controller	Set	Succ. (%)	$\overline{h}_{\text{drop}}$ [m]	$\overline{t}_{\text{hold}}$ [s]	$\overline{\lvert v_{x,\text{hold}}\rvert}$ [m/s]	$\overline{\lvert v_{y,\text{hold}}\rvert}$ [m/s]	$\overline{\lvert v_{z,\text{hold}}\rvert}$ [m/s]
PID	1	100	8.29	1.57	1.34	2.12	0.15
	2	90	20.85	2.76	2.73	1.58	0.2
	3	90	21.95	3.5	3.97	2.35	0.21
	4	70	22.4	7.5	6.01	5.36	0.7
	5	90	15.87	3.32	0.36	1.65	0.1
	6	100	11.64	3.53	2.26	2.9	0.25
	7	60	16.54	5.2	4.7	2.9	0.55
NMPC	1	100	5.74	1.27	0.19	0.15	0.07
	2	70	12.8	2.82	0.06	0.22	0.12
	3	80	9.96	3.1	0.35	1.35	0.22
	4	50	12.3	8.70	3.25	3.85	0.43
	5	80	17.36	6.07	0.24	0.42	0.08
	6	50	10.08	6.68	0.06	0.34	0.08
	7	40	18.9	6.32	0.4	0.6	0.12

the unmodeled dynamics have greater effect on the vehicle. This effect becomes more pronounced in the NMPC controller, as it relies heavily on the vehicle model, whereas the PID controller does not. Additionally, the average time to hold the position $\overline{t}_{hold}$ becomes greater as the wind disturbances increase, and so does the average of the absolute values of the velocity in the z direction. For successful recoveries, the height drop using the NMPC controller is smaller in all the cases presented for sets 1-4.

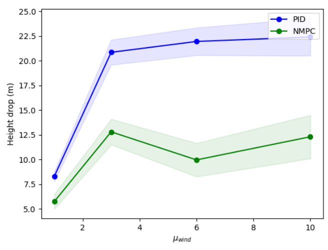
Sets 5-7 consider a high initial velocity of the drone in the y direction and some wind disturbance. It is seen that the PID controller is able to stabilize the the UAV in almost all simulations, while the NMPC fails more as the wind disturbances increase. The NMPC also takes more time the achieve a successful recovery attempt when the vehicle's initial velocity is considerable. This is also true for the simulation without wind disturbances.

For all simulations, it is seen that the average of the absolute velocities in the x and y axes, $\overline{\lvert v_{x,\text{hold}}\rvert}$, $\overline{\lvert v_{y,\text{hold}}\rvert}$, respectively, are smaller for the NMPC controller. Possibly, this can be explained by the fact the the NMPC actively has penalization on the linear velocities in these axes, whereas the PID only attempts to stabilize the height, and therefore, the linear velocity in the z axis.

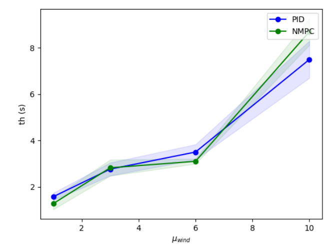
In general, it is possible to conclude that PID controller implemented is more robust to unmodeled dynamics than the NMPC. This can be seen in the simulations with



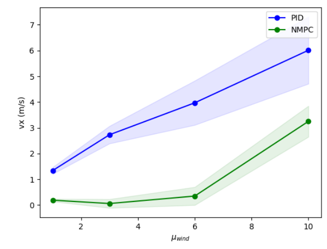
(a) Success.



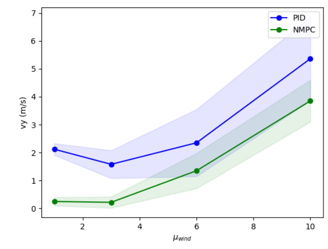
(b) $\overline{h}_{drop}$.



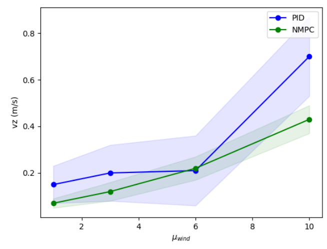
(c) $\overline{t}_{hold}$.



(d) $\overline{|v_{x,hold}|}$.

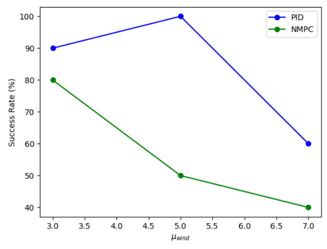


(e) $\overline{|v_{y,hold}|}$.

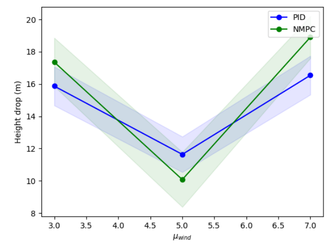


(f) $\overline{|v_{z,hold}|}$.

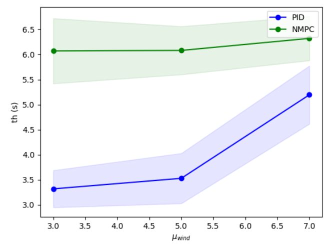
Figure 6.5: Results for including wind disturbances with small initial velocity.



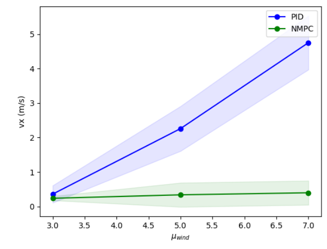
(a) Success.



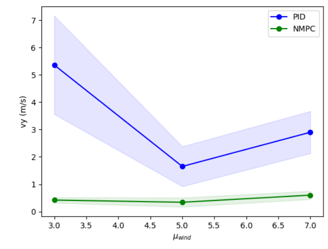
(b) $\overline{h}_{drop}$.



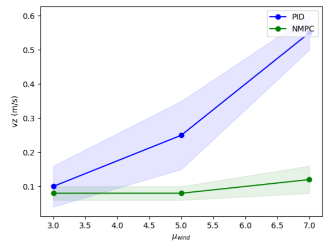
(c) $\overline{t}_{hold}$.



(d) $\overline{|v_{x,hold}|}$.



(e) $\overline{|v_{y,hold}|}$.



(f) $\overline{|v_{z,hold}|}$.

Figure 6.6: Results for including wind disturbances with high initial velocity.

considerable wind perturbations and with a high initial velocity. Due to the fact that MPC relies heavily on the model of the system being controlled, introducing dynamics that the model does not account for leads to inaccurate predictions of the system's future behavior. The PID on the other hand does not rely so heavily on the model and in the simulations performed seems to be more robust. However, when the disturbances introduced are not that significant, for instance when the wind disturbances are not high, the NMPC leads to a recovery with minimum height lost and minimum time, while still respecting the limits of thrust and torque. Since the implemented NMPC considers the velocities and positions in all axes and not only on the z axis, it can reduce the lateral variation of the vehicle. Finally, as previously mentioned, the NMPC is not yet capable of running in real-time, operating at an average frequency of 20 Hz, which indicates that further optimization is still necessary.

Chapter 7

Conclusion and Future Work

The work described in this thesis aimed to design recovery controller for a quadrotor tailsitter UAV that is launched in the air with unknown initial attitude and velocity. Two different approaches were presented, one relying of PID controllers and the other on NMPC. In both, the attitude is represented by unit quaternions and the error quaternion is divided into two different components, one corresponding to the inclination error and the other to the heading to increase the robustness against large attitude errors. Only after the vehicle is relatively close to the nominal attitude there is an attempt to control the heading of the vehicle. The performance of the controllers were evaluated through SITL simulations. It is shown that for small wind perturbations and low initial velocity, the NMPC is able to recover the vehicle in less time and with lower height drop than the PID, while still respecting the limits of thrust and torques. On the other hand, the PID is more robust to unmodeled introduced by wind disturbances and is computationally faster, allowing for real time simulation. Additionally, since the NMPC formulation contains penalties on the velocity over all three axes, it is seen that in all simulations the final velocity in the x and y axes is always lower than when using the PID, which only accounts for velocities in the z axis.

In future work we plan to conduct experiments to develop a more accurate aerodynamic model, for instance by performing wind tunnel experiments of the real UAV, as well as assess the performance of the controllers in the real world. In addition, we plan to increase the performance of the PID controller by replacing the altitude controller to another that can take into account the positions and velocities in all three axes. Finally, we want to increase the computational efficiency of the NMPC.

Bibliography

[1] Z. Zaheer, A. Usmani, E. Khan, and M. A. Qadeer, “Aerial surveillance system using uav,” in *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, 2016, pp. 1–7.

[2] C. A. Thiels, J. M. Aho, S. P. Zietlow, and D. H. Jenkins, “Use of unmanned aerial vehicles for medical product transport,” *Air Medical Journal*, vol. 34, no. 2, pp. 104–108, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1067991X14003332

[3] Y. Naidoo, R. Stopforth, and G. Bright, “Development of an uav for search & rescue applications,” in *IEEE Africon ’11*, 2011, pp. 1–6.

[4] S. Asadzadeh, W. J. de Oliveira, and C. R. de Souza Filho, “Uav-based remote sensing for the petroleum industry and environmental monitoring: State-of-the-art and perspectives,” *Journal of Petroleum Science and Engineering*, vol. 208, p. 109633, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920410521012675

[5] F. Zhang, X. Lyu, Y. Wang, H. Gu, and Z. Li, *Modeling and Flight Control Simulation of a Quadrotor Tailsitter VTOL UAV.* American Institute of Aeronautics and Astronautics, 2017, pp. 1–13. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2017-1561

[6] A. S. Saeed, A. B. Younes, C. Cai, and G. Cai, “A survey of hybrid unmanned aerial vehicles,” *Progress in Aerospace Sciences*, vol. 98, pp. 91–105, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042117302233

[7] S. Verling, B. Weibel, M. Boosfeld, K. Alexis, M. Burri, and R. Siegwart, “Full attitude control of a vtol tailsitter uav,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3006–3012.

[8] PX4 Autopilot Development Team, “PX4 Autopilot Documentation,” https://docs.px4.io/main/en/frames_vtol/, 2023, [Accessed on: December 18, 2023].

[9] S. Panza, M. Sato, M. Lovera, and K. Muraoka, “Robust attitude control design of quad-tilt-wing uav: A structured mu-synthesis approach,” *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 781–786, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:53099841

[10] P. Casau, D. Cabecinhas, and C. Silvestre, “Hybrid control strategy for the autonomous transition flight of a fixed-wing aircraft,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2194–2211, 2013.

[11] J. Zhou, X. Lyu, Z. Li, S. Shen, and F. Zhang, “A unified control method for quadrotor tail-sitter uavs in all flight modes: Hover, transition, and level flight,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 4835–4841.

[12] J. Diebel, “Representing attitude: Euler angles, unit quaternions, and rotation vectors,” *Matrix*, vol. 58, 01 2006.

[13] N. A. Chaturvedi, A. K. Sanyal, and N. H. McClamroch, “Rigid-body attitude control,” *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 30–51, 2011.

[14] M. D. Shuster *et al.*, “A survey of attitude representations,” *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[15] V. Pesce, A. Colagrossi, and S. Silvestrini, *Modern Spacecraft Guidance, Navigation, and Control: From System Modeling to AI and Innovative Applications.* Elsevier, 11 2022.

[16] J. Solà, “Quaternion kinematics for the error-state kf,” *arXiv preprint arXiv:1711.02508*, 03 2015.

[17] J. M. Lemos, *Controlo no Espaço de Estados*, 1st ed. IST press, 2019.

[18] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, “Review on model predictive control: an engineering perspective,” *The International Journal of Advanced Manufacturing Technology*, vol. 117, pp. 1–23, 11 2021.

[19] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," *21st Benelux Meeting on Systems and Control*, 01 2002.

[20] X. Yang, G. Liu, A. Li, and L. Van Dai, "A predictive power control strategy for dfigs based on a wind energy converter system," *Energies*, vol. 10, p. 1098, 07 2017.

[21] J. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 38–52, 2000.

[22] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza, "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1722–1729.

[23] G. Dicker, F. Chui, and I. Sharf, "Quadrotor collision characterization and recovery control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5830–5836.

[24] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on so(3)," in *2015 IEEE Conference on Control Applications (CCA)*, 2015, pp. 1160–1166.

[25] K. Nguyen, S. Schoedel, A. Alavilli, B. Plancher, and Z. Manchester, "Tinympc: Model-predictive control on resource-constrained microcontrollers," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[26] F. Nan, S. Sun, P. Foehn, and D. Scaramuzza, "Nonlinear mpc for quadrotor fault-tolerant control," *IEEE Robotics and Automation Letters*, vol. 7, 04 2022.

[27] R. Ritz and R. D'Andrea, "A global controller for flying wing tailsitter vehicles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2731–2738.

[28] E. A. Tal and S. Karaman, "Global trajectory-tracking control for a tailsitter flying wing in agile uncoordinated flight," in *AIAA Aviation 2021 Forum*, 2021, p. 3214.

[29] L. Ribeiro Lustosa, F. Defaÿ, and J.-M. Moschetta, "Global singularity-free aerodynamic model for algorithmic flight control of tail sitters," *Journal of Guidance, Control, and Dynamics*, vol. 42, pp. 1–14, 12 2018.

[30] T. Matsumoto, K. Kita, R. Suzuki, A. Oosedo, K. Go, Y. Hoshino, A. Konno, and M. Uchiyama, "A hovering control strategy for a tail-sitter vtol uav that increases stability against large disturbance," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 54–59.

[31] X. Lyu, H. Gu, J. Zhou, Z. Li, S. Shen, and F. Zhang, "A hierarchical control approach for a quadrotor tail-sitter vtol uav and experimental verification," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5135–5141.

[32] B. Li, W. Zhou, J. Sun, C.-Y. Wen, and C.-K. Chen, "Development of model predictive controller for a tail-sitter vtol uav in hover flight," *Sensors*, vol. 18, no. 9, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/9/2859

[33] B. Li, W. Zhou, J. Sun, C. Wen, and C. Chen, "Model predictive control for path tracking of a vtol tailsitter uav in an hil simulation environment," in *2018 AIAA modeling and simulation technologies conference*, 2018, p. 1919.

[34] G. Lu, Y. Cai, N. Chen, F. Kong, Y. Ren, and F. Zhang, "Trajectory generation and tracking control for aggressive tail-sitter flights," *The International Journal of Robotics Research*, vol. 43, no. 3, pp. 241–280, 2024. [Online]. Available: https://arxiv.org/abs/2212.11552

[35] S. H. Mathisen, T. I. Fossen, and T. A. Johansen, "Non-linear model predictive control for guidance of a fixed-wing uav in precision deep stall landing," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015, pp. 356–365.

[36] S. Mathisen, K. Gryte, S. Gros, and T. Johansen, "Precision deep-stall landing of fixed-wing uavs using nonlinear model predictive control," *Journal of Intelligent & Robotic Systems*, vol. 101, 01 2021.

[37] M. Allenspach and G. J. J. Ducard, "Nonlinear model predictive control and guidance for a propeller-tilting hybrid unmanned air vehicle," *Automatica*, vol. 132, p. 109790, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109821003101

[38] A. Vuruskan, B. Yuksek, U. Ozdemir, A. Yukselen, and G. Inalhan, "Dynamic modeling of a fixed-wing vtol uav," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 483–491.

[39] E. Smeur, Q. Chu, G. De Croon, B. Remes, C. De Wagter, and E. Van der Horst, "Modelling of a hybrid uav using test flight data," in *International micro air vehicle competition and conference*, vol. 201, 2014, pp. 4–203.

[40] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[41] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[42] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control. technical report," Eidgenössische Technische Hochschule Zürich, Departement Maschinenbau und Verfahrenstechnik, Zürich, Report, 2013.

[43] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.